ESCUELA SUPERIOR DE TECNOLOGÍA Y CIENCIAS EXPERIMENTALES

MÁSTER UNIVERSITARIO EN INGENIERÍA INDUSTRIAL

DESARROLLO DE UNA APLICACIÓN PARA LA OPTIMIZACIÓN ENERGÉTICA DE PLANTAS DE DEPURACIÓN DE AGUA CON INTERFAZ DE USUARIO

TRABAJO FINAL DE MÁSTER

AUTOR

JOSE CANDEA ROMÁN

DIRECTOR DEL PROYECTO

IGNACIO PEÑARROCHA ALÓS

ROBERTO SANCHIS LLOPIS

CASTELLÓN, JULIO DEL 2019

AGRADECIMIENTOS

A Ignacio Peñarrocha, por el tiempo dedicado y el apoyo mostrado en todo momento, transformando este proyecto en un reto personal.

A Aroa Galán, por ser parte indispensable de este proyecto, por ser revisora, consejera y compañera, por convertir esto en algo especial. A caminar.

A mi madre y a mi hermana, por hacer fuerza en estos últimos meses para superar lo que sea y aguantar mis altibajos sin saber muy bien el motivo.

MUCHAS GRACIAS.

ÍNDICE GENERAL

I. MEMORIA

ÍNDICE DE CAPÍTULO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

I.1 OBJETO

I.1.1 OBJETIVOS

El objetivo principal de este proyecto es desarrollar y aplicar un algoritmo de control que permita la optimización, tanto energética como económica, del funcionamiento de cualquier Estación Depuradora de Aguas Residuales (EDAR). Además, se fija también como objetivo la realización de una interfaz de usuario que permita modificar y/o adaptar la aplicación a las condiciones de uso deseadas.

I.1.2 JUSTIFICACIÓN

Este proyecto surge con la intención de facilitar la implantación y aplicación de algoritmos de optimización a las EDAR. Además, su importancia reside en la necesidad de desarrollar un algoritmo que se pueda a aplicar a diferentes tipos de plantas independientemente de sus características y condiciones de funcionamiento.

I.2 ALCANCE

El presente proyecto se encuadra en el ámbito del control de sistemas, más concretamente en la rama de control predictivo y optimización de recursos. En concreto, se implementarán diferentes técnicas tanto de identificación, como de control y optimización.

Si bien se simula la planta descrita en el *Benchmark Simulation Model nº1* (BSM1) de J. Álex, L. Benedetti et al., la conexión con el software de cálculo y modelado de EDAR permite diseñar y simular cualquier tipo de estación de depuración.

Además, el proyecto no termina con el desarrollo de la estrategia de control, sino que pretende realizar una aplicación distribuible que ponga a disposición del usuario final una serie de funcionalidades que faciliten la implantación y monitorización de las plantas reales.

I.3 ANTECEDENTES

En la actualidad, el consumo energético es uno de los principales problemas a afrontar si se quiere dirigir la sociedad hacia un desarrollo sostenible. De hecho, hay una gran cantidad de procesos que funcionan en continuo y que contribuyen considerablemente al consumo de una ciudad. Por este motivo, es verdaderamente importante el control y optimización de estos sistemas. Este proyecto se centra en la optimización de uno de los sistemas más necesarios de las ciudades actuales, las Estaciones De Depuración De Aguas Residuales (EDAR). En una sociedad que cada vez se concentra más en las grandes ciudades concentrando de igual forma las aguas residuales, es necesario que el control del sistema de depuración sea lo más óptimo posible. Sobre este tema se han realizado numerosas investigaciones y se ha llegado a diferentes conclusiones que serán estudiadas y consideradas en el desarrollo de este proyecto.

I.4 NORMAS Y REFERENCIAS

I.4.1.- DISPOSICIONES LEGALES Y NORMAS APLICADAS

- **ORDENANZA MUNICIPAL REGULADORA DE VERTIDOS LÍQUIDOS RESIDUALES**, de Castellón de la Plana.
- **UNE 157001-2014,** de junio de 2014 "*Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico*".
- **ISO 13849-1:2015**, de diciembre de 2015 "*Safety of machinery -- Safety-related parts of control systems -- Part 1: General principles for design*"

I.4.2.- PROGRAMAS DE CÁLCULO

Para la implementación de las estrategias de control, la optimización y el desarrollo de la aplicación se ha utilizado:

- Pack **MATLAB** 2019a
    - MATLAB
    - MATLAB Compiler
    - App Designer
    - YALMIP + *Solver*

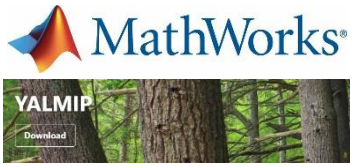


Figura 1.- Logo de Pack Matlab y Yalmip

Para el modelado y cálculo de las ecuaciones propias de una EDAR cualquiera, además de motor de cálculo para el simulador se han utilizado los siguientes softwares:

- **WEST** – MIKE by DHI
    - WEST for DESIGN
    - WEST for OPERATORS
    - WEST for OPTIMIZATION

Figura 2.- Logo Mike by DHI

- **TORNADO** SHELL (ventana de comando de WEST)

I.4.3.- PLAN DE GESTIÓN DE LA CALIDAD DEL PROYECTO

Puesto que una parte considerable de el proyecto consiste en la comparación y análisis de resultados, es de vital importancia que estos procesos se realicen con la mayor precisión y de la forma más meticulosa y ordenada posible.

Además, al tratarse de un proyecto con diferentes fases y sub-fases, se establecen una serie de controles de calidad y *check-points* intermedios. Estos tienen el objetivo de determinar una posible causa de fallo en el desarrollo en la siguiente fase, incluso antes de que se produzca. Por ejemplo, los resultados de la FASE 1 deben tener la calidad exigida antes de que se conviertan en entrada de la FASE 2.

I.4.4.- BIBLIOGRAFÍA

[1] S. Garrido Directores and L. Moreno Carlos Balaguer, “Identificación, Estimación y Control de Sistemas No-lineales mediante RGO,” 1999.

[2] S. J. Qin and T. A. Badgwell, “AN OVERVIEW OF INDUSTRIAL MODEL PREDICTIVE CONTROL TECHNOLOGY.”

[3] F. Claeys, D. J. De Pauw, L. Benedetti, and P. A. Vanrolleghem, “Tornado: A versatile and efficient modelling & virtual experimentation kernel for water quality systems,” in *Proceedings of iEMSs 2006*, 2006.

[4] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, “Identification of Hammerstein–Wiener models,” *Automatica*, vol. 49, no. 1, pp. 70–81, Jan. 2013.

[5] J. K. Gruber and C. Bordons, “Control predictivo no lineal basado en modelos de volterra. aplicación a una planta piloto,” *Rev. Iberoam. Automática e Informática Ind. RIAI*, vol. 4, no. 3, pp. 34–45, Jul. 2007.

[6] R. Saagi, X. Flores-Alsina, S. Kroll, K. V. Gernaey, and U. Jeppsson, “A model library for simulation and benchmarking of integrated urban wastewater systems,” *Environ. Model. Softw.*, vol. 93, pp. 282–295, Jul. 2017.

[7] J.-W. Chen and J. Zhang, “Comparing Text-based and Graphic User Interfaces for novice and expert users.,” *AMIA ... Annu. Symp. proceedings. AMIA Symp.*, vol. 2007, pp. 125–9, Oct. 2007.

[8] J. K. Gruber and I. Peñarrocha, “SOLUCIONES ALTERNATIVAS AL CONTROL PREDICTIVO BASADO EN MODELO DE VOLTERRA.”

[9] “Computation Visualization Programming For Use with MATLAB ® MATLAB Web Server MATLAB Web Server,” 1999.

[10] I. Peñarrocha*. "Apuntes de la asignatura SJA010 - Automatización y control avanzado de procesos", 2018*

[11] Alex, J., et al. "Benchmark simulation model no. 1 (BSM1)." Report by the IWA Taskgroup on Benchmarking of Control Strategies for WWTPs. 2008. (Corrección de 2018)

[12] C. Bordons, “Control Predictivo: Metodologia, tecnologia y nuevas perspectivas. I Curso de Especialización en Automática,” *Aguadulce, Almeria*. 2000.

I.4.5.- OTRAS REFERENCIAS

Además de la documentación recogida en el apartado anterior, se han consultado las siguientes referencias:

API El tiempo – descarga de información climatológica – https://www.tiempo.com/api/

API ESIOS – Página de REE para conocer tarifas – https://www.esios.ree.es/es/pagina/api

WEST – Documentación uso software – https://www.mikepoweredbydhi.com/products/west

STOAT – Programa de simulación – http://www.wrcplc.co.uk/ps-stoat

GPS-X – Programa de simulación y modelación – https://www.hydromantis.com/GPSX.html

YALMIP – Optimizador – https://yalmip.github.io/tutorials/

SOLVERS – Complementos para YALMIP:

- ***SDPT3*** version 4.0 – http://www.math.nus.edu.sg/~mattohkc/sdpt3.html
- ***MOSEK*** – https://www.mosek.com/
- ***PENBMI*** – http://www.penopt.com/penbmi.html
- **SeDuMi** – http://sedumi.ie.lehigh.edu/

MATLAB – Descarga y documentación – https://es.mathworks.com/

CVX – Optimizador de LMI - http://cvxr.com/cvx/

I.5 DEFINICIONES Y ABREVIATURAS

A continuación, se presentan las definiciones y abreviaturas recopiladas en función de la parte del proyecto en la que aparecen.

I.5.1.- SOBRE EL SISTEMA

AMONIO. NH_4. Catión poliatómico procedente de la reacción entre el amoníaco y con los átomos de hidrogeno. En este proyecto, constituirá uno de los límites de las salidas y por tanto un requisito de diseño.

ASM1. ACTIVATED SLUDGE MODEL nº 1, o Modelo de Lodos Activados número 1. Estas son un conjunto de ecuaciones que se utilizan para simular o modelar el comportamiento de las EDAR.

BSM1. El BENCHMARK SIMULATION MODEL nº 1, o modelo de referencia número 1, consiste en unas series de ecuaciones (entre las que se incluye el ASM1), restricciones y condiciones que modelizan el funcionamiento del tratamiento secundario de una EDAR. Incluye también una serie de experimentos que sirven para verificar el correcto funcionamiento de la simulación [6].

EDAR/WWTP. ESTACIÓN de DEPURACIÓN de AGUAS RESIDUALES/ WASTEWATER TREATEMENT PLANT. Una EDAR es el lugar en que se recogen y depuran las aguas residuales procedentes de núcleos urbanos, con el fin de reducir los niveles de contaminantes y sustancias en suspensión hasta llegar a los límites admisibles.

$\boldsymbol{K_La}$ – Coeficiente volumétrico de transferencia de oxígeno. En concreto hace referencia a la facilidad y posibilidad de transferencia líquido-gas. En este proyecto, esta variable se tomará como proporcional al consumo del soplador que introduce el aire en tanque de agitación.

I.5.2.- SOBRE BSM1

Para conocer la terminología con la que trabaja el documento se presentan las siguientes tablas y figuras en las que se especifican el significado de las abreviaturas y sus unidades.

NOMBRE	*NOTACIÓN*	*UNIDADES*
SOLUBLE INERT ORGANIC MATTER	S_{I}	$g\ COD \cdot m^{-3}$
READILY BIODEGRADABLE SUBSTRATE	S_S	$g\ COD \cdot m^{-3}$
PARTICULATE INERT ORGANIC MATTER	X_I	$g\ COD \cdot m^{-3}$
SLOWLY BIODEGRADABLE SUBSTRATE	X_S	$g\ COD \cdot m^{-3}$
ACTIVE HETEROTROPHIC BIOMASS	$X_{B,H}$	$g\ COD \cdot m^{-3}$
ACTIVE AUTOTROPHIC BIOMASS	$X_{B,A}$	$g\ COD \cdot m^{-3}$
PARTICULATE PRODUCTS ARISING FROM BIOMASSDECAY	X_P	$g\ (-COD) \cdot m^{-3}$
NITRATE AND NITRITE NITROGEN	S_{NO}	$g\ \mathrm{N} \cdot m^{-3}$
OXYGEN	S_O	$g\ \mathrm{N} \cdot m^{-3}$
NH_4^++ NH_3 NITROGEN	S_{NH}	$g\ \mathrm{N} \cdot m^{-3}$
SOLUBLE BIODEGRADABLE ORGANIC NITROGEN	S_{ND}	$g\ \mathrm{N} \cdot m^{-3}$
PARTICULATE BIODEGRADABLE ORGANIC NITROGEN	X_{ND}	$g\ \mathrm{N} \cdot m^{-3}$
ALKALINITY	S_{ALK}	$\mathrm{mol} \cdot m^{-3}$

Tabla 1.- Listado de las variables del ASM1 – BSM1

Al haber una gran cantidad de caudales a tener en cuenta, a continuación, se muestra un esquema y un listado con sus descripciones.

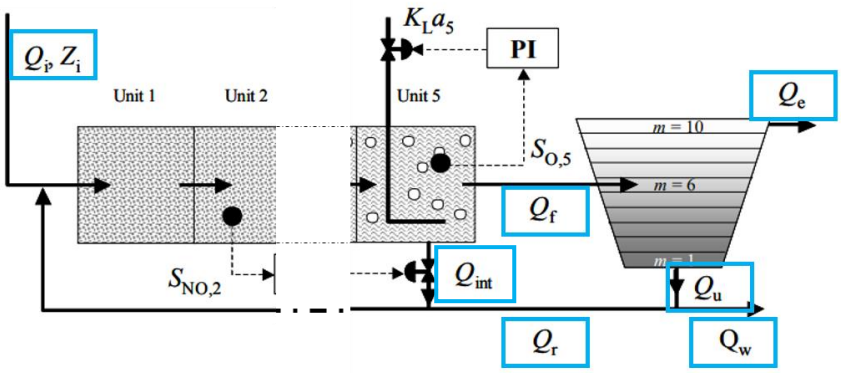


Figura 3.- Representación de las variables de los caudales

NOMBRE	NOTACIÓN
FLUJO DE ENTRADA	Q_0
FLUJO DE ALIMENTACIÓN AL DECANTADOR (FEEDFLOW)	Q_f
CAUDAL DE RECIRCULACIÓN INTERNA	Q_{int}
CAUDAL EFLUENTE	Q_e
FLUJO INFERIOR (UNDERFLOW)	Q_u
CAUDAL DE RECIRCULACIÓN EXTERNA	Q_r
FLUJO DE PURGA (WASTAGE)	Q_w

Tabla 2.- Listado de caudales del sistema

Por su lado las diferentes etapas se definen en las siguientes líneas:

ANOXIC SECTION.- Un proceso anóxico es aquel que se produce en un medio con ausencia de oxígeno, en este caso hace referencia a los tanques en los que no se introduce aireación. En este tipo de sistemas, el proceso metabólico da como resultado la transformación del nitrógeno de la molécula de nitrato en nitrógeno gas.

AERATED SECTION.- Por contra al caso anterior, la sección aireada hace referencia a aquellos tanques en los que se está introduciendo aire (K_La). En esta fase se produce un crecimiento de las bacterias que permiten reducir los niveles de amonio.

ASU – ACTIVE SLUDGE UNIT.- Unidades o tanques en los que se produce el proceso de *Activated sludge* que consiste en el tratamiento de aguas residuales mediante aireación y reacciones químicas.

NOMBRE	FASE
Unit 1 – ASU1 – ACTIVATED SLUDGE UNIT (1000 m^3)	Anoxic section
Unit 2 – ASU2 – ACTIVATED SLUDGE UNIT (1000 m^3)	Anoxic section
Unit 3 – ASU3 – ACTIVATED SLUDGE UNIT(1333 m^3)	Aerated section
Unit 4 – ASU4 – ACTIVATED SLUDGE UNIT (1333 m^3)	Aerated section
Unit 5 – ASU5 – ACTIVATED SLUDGE UNIT (1333 m^3)	Aerated section

Tabla 3.- Listado de tanques (BSM1)

En la siguiente figura se observan la ordenación de los tanques del sistema, la L en negro representa el soplador.

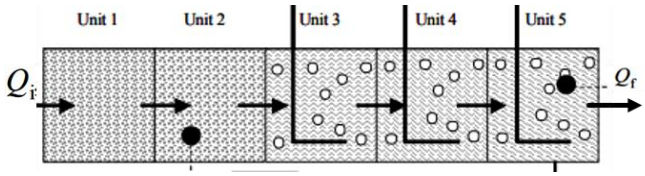


Figura 4.- Esquema de los 5 tanques que forman el sistema

Finalmente, se encuentra el decantador o ***secondary clarifier,*** su definición y parámetros más importantes son:

SECONDARY CLARIFIER – DECANTADOR: Se trata de un depósito circular en el que el fango activo reposa durante un periodo de tiempo suficiente para que vaya produciendo la sedimentación de la biomasa. Este está formado por capas, en este caso 10 (**m**=1…10).

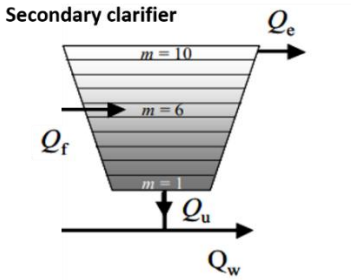


Figura 5.- Esquema del decantador (10 capas)

En el **ANEXO 1: EXTRACTO BSM1 (FÓRMULAS)** se amplía la información sobre el sistema presentando un resumen de la formulación interna.

I.5.3.- SOBRE CONTROL

MPC – MODEL based PREDICTIVE CONTROL. Es una estrategia de control de procesos y sistemas que cuya base es redecir el efecto sobre la salida de las acciones de control, actuales y futuras. Su peculiaridad es que para hacer esta predicción utiliza un modelo del sistema.

GPC – GENERALIZED PREDICTIVE CONTROL. Es una estrategia de control que esencialmente se basa en horizontes finitos sin requisitos para sistemas muy concretos. Por lo tanto, se puede considerar obsoleto y sustituido por el MPC.

PID – PROPORCIONAL, INTEGRAL Y DERIVATIVO. Son las siglas del controlador formado por la realimentación de los tres tipos de controles que indica su nombre.

OL – OPEN LOOP. Se refiere a cuando un sistema no es alimentado con su salida y por lo tanto no tiene información sobre ella.

SDP – SEMIDEFINITE PROGRAMMING. Es un campo de la optimización convexa relacionado con la optimización de una función objetivo lineal sobre la intersección de conos de matrices positivas semidefinidas.

LMI – LINEAR MATRIX INEQUALITY. Es un término utilizado en la optimización convexa y se utiliza para expresar condiciones sobre un sistema matricial.

n – Orden del modelo. Hace referencia a cuantos datos precedentes utiliza el modelo.

Delay – Retardo. Hace referencia al tiempo que pasa entre un cambio de una variable y su efecto sobre el sistema.

I.5.4.- SOBRE DISEÑO DE APLICACIONES

API – APPLICATION PROGRAMMING INTERFACE. Es un programa que permite a las aplicaciones comunicarse con otras. Normalmente, una API es un programa público basado en la web, que tras una petición devuelve datos en formato JSON o XML.

GUI – GRAPHICAL USER INTERFACE. Es un programa informático que permite al usuario interactuar con el código de forma simple y utilizando imágenes y objetos gráficos.

JSON – JAVASCRIPT OBJECT NOTATION. Es un formato de texto sencillo que facilita el intercambio de datos.

XML – EXTENSIBLE MARKUP LANGUAGE. Es un lenguaje de marcas o esquema que utiliza un estándar y siguiendo una estructura que permite encapsular la información.

I.5.5.- SOBRE MATLAB

MATLAB – MATRIX LABORATORY. Es un entorno computacional y de programación basado en el cálculo matricial que cuenta con lenguaje de desarrollo propio. En el documento se hace referencia a él como *Matlab* para facilitar la lectura.

MEX – MATLAB EXECUTABLE. Son funciones creadas en Matlab que llaman a subrutinas de C, C++ o Fortan.

TOOLBOX –"CAJA DE HERRAMIENTAS". Es un conjunto de funciones diseñadas para un propósito relacionado y vendidas o distribuidas en un paquete.

I.6 REQUISITOS DE DISEÑO

I.6.1.- REQUISITOS DE DISEÑO DEL SIMULADOR - BSM1

Para la realización del proyecto se partirá de los datos proporcionados por el documento BSM1. siendo esté quien fije los requisitos de la parte de simulación. Considerando como valido aquel modelo que consiga resultados iguales a los reflejados en BSM1. En los puntos siguientes, se presentan los datos que han de cumplirse y las condiciones que deben utilizarse de partida.

Por otro lado, para que el simulador sea extrapolable a otro tipo de EDAR debe ser capaz de alcanzar los mismos resultados exactos que WEST al realizar simulaciones.

I.6.1.1.- BSM1 – PREMISAS GENERALES

Puesto que el presente proyecto se basa en lo que se expone en el *Benchmark* los requisitos de diseño viene marcados por este documento. Para empezar la planta se modelará siguiendo las premisas marcada en el documento, en la siguiente figura se presenta el esquema general de la misma.

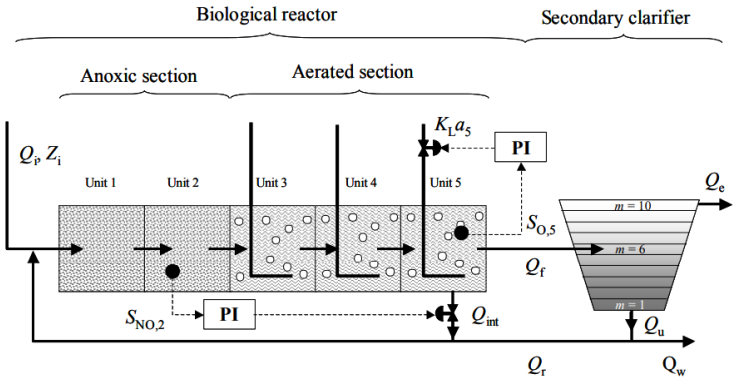


Figura 6.- Esquema general de la planta

En este documento se exponen y explican las fórmulas y relaciones que definen el ASM1, mostrando así el modelo del bioproceso que rigen este sistema.

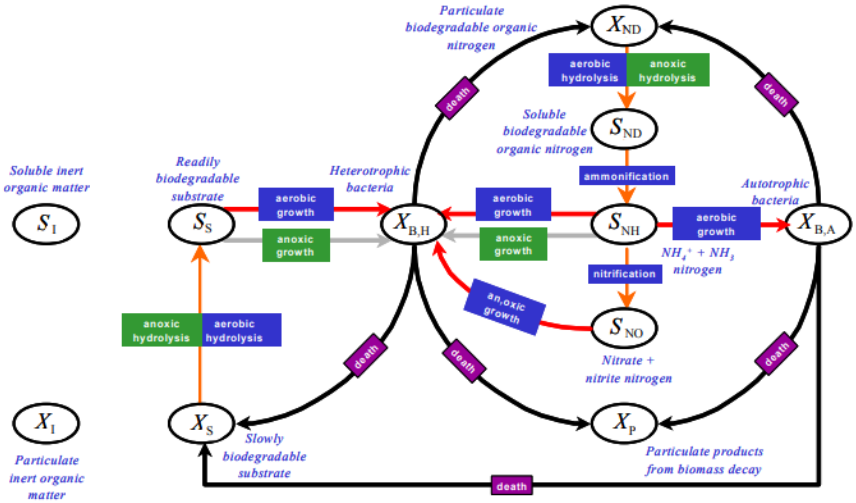


Figura 7.- Esquema general del ASM1

Así mismo en el documento se recogen todas las condiciones iniciales que se deben respetar para llevar a cabo la simulación, por lo tanto, es de vital importancia revisar y

comprender la información que recoge. En el apartado de **abreviaturas y definiciones** se muestra una tabla con las variables más representativas del modelo. Por otro lado, dispone de tres ficheros de texto en los que se encuentran los datos de entrada en condiciones distintas, estos datos se analizarán en profundidad en el siguiente apartado.

El BSM1 es el documento idóneo para realizar una simulación estandarizada ya que dentro de la comunidad científica está muy extendido y existen numerosos artículos e informes con los que comparar resultados.

I.6.1.2.- DATOS DE ENTRADA

Como se ha comentado precedentemente, para la realización de este proyecto se tomarán los datos suministrados junto al BSM1, estos serán los datos que se introducirán en el simulador para reproducir las variables de salida de una EDAR real. En concreto, el análisis se centrará en el fichero con nombre **Inf_dry_2006.txt** que recoge los datos de un día seco o sin precipitaciones. A continuación, se presentan graficadas las dos variables más importantes del fichero de entrada (Q total de entrada y Amonio entrante).

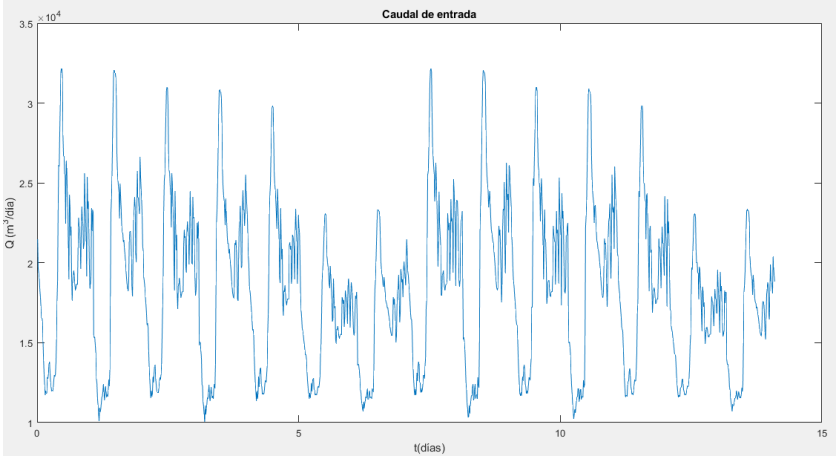


Figura 8.- Caudal de entrada en día seco

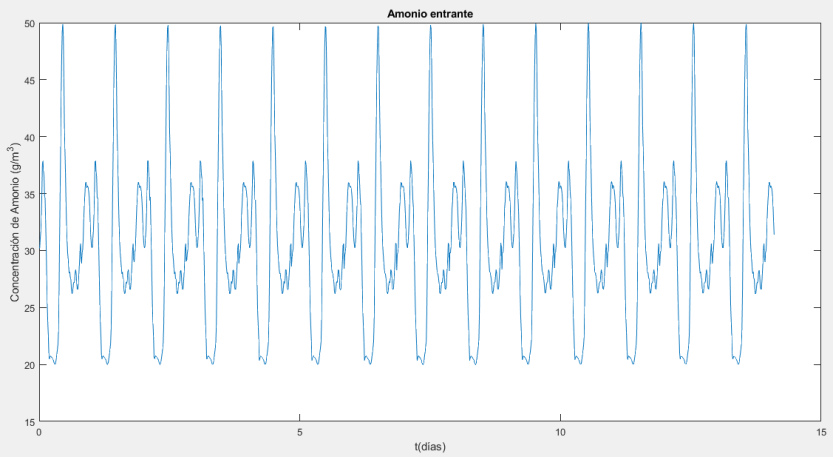


Figura 9.- Amonio entrante

La estructura del fichero de partida se presenta en la siguiente tabla. Como se puede observar cuanta con todas las variables que determinarán el cambio y comportamiento de la planta. Además, este fichero presenta datos de 14 días.

t	H_2O	S_{ALK}	S_I	S_{ND}	S_{NH}	S_{NO}	S_O	S_S	X_{BA}	...	X_S
d	m3/d	g/m3	g/m3	g/m3	g/m3	g/m3	g/m3	g/m3	g/m3	...	g/m3
0	21477	30	30	6.247	34.679	0.001	0.001	62.465	0.001	...	224.352
0.01042	21474	30	30	5.374	32.066	0.001	0.001	53.741	0.001	...	224.324
0.02083	19620	30	30	5.444	29.899	0.001	0.001	54.448	0.001	...	224.373
...											
...											
13.98958	18409	30	30	6.283	34.387	0.001	0.001	62.838	0.001	...	200.958
14	21477	30	30	6.246	34.679	0.001	0.001	62.465	0.001	...	224.352

Tabla 4.- Resumen datos de "Municipality" del BSM1

Para observar a simple vista todas las variables se han representado en el siguiente gráfico.

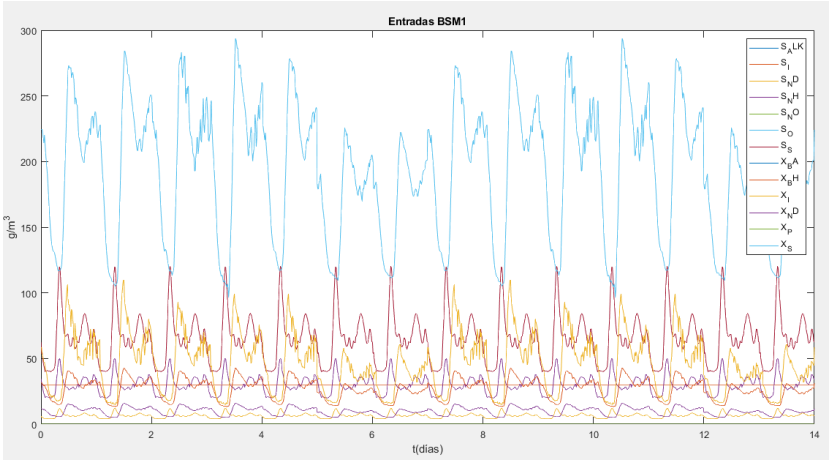


Figura 10.- Representación señales de entrada

Como se observa en las representaciones los valores de entrada son apenas comparables, por lo que se ha decidido aplicar una normalización sobre cada una de las variables. Haciendo que todas se encuentren entre 0 y 1. Esto es importante sobre todo a la hora de comparar el caudal con cualquier otra salida. En la siguiente imagen se ilustra el proceso de normalización, así como la fórmula utilizada:

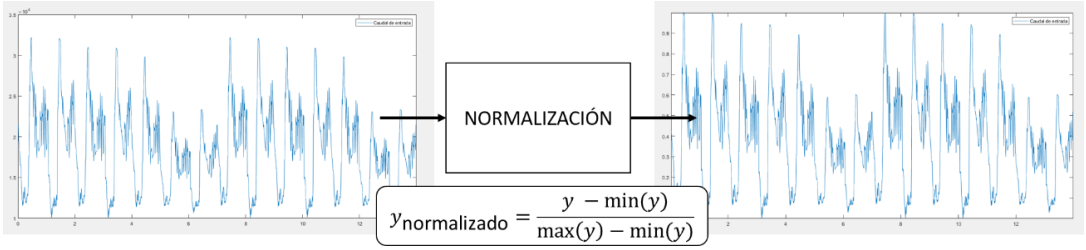


Figura 11.- Esquema de normalización de señales

Si observamos los datos de entrada de una forma más concreta se pueden extraer conclusiones interesantes. En las siguientes figuras se presenta la evolución intradía e intrasemana del caudal entrante. En la primera se muestra un gran pico de caudal alrededor de las 10.30 de la mañana, mientras que en la segunda se ve claramente la diferencia de perfil entre un día entresemana y un día de fin de semana. Esta diferencia se debe en gran parte a que los fines de semana se reduce de una manera considera la actividad industrial

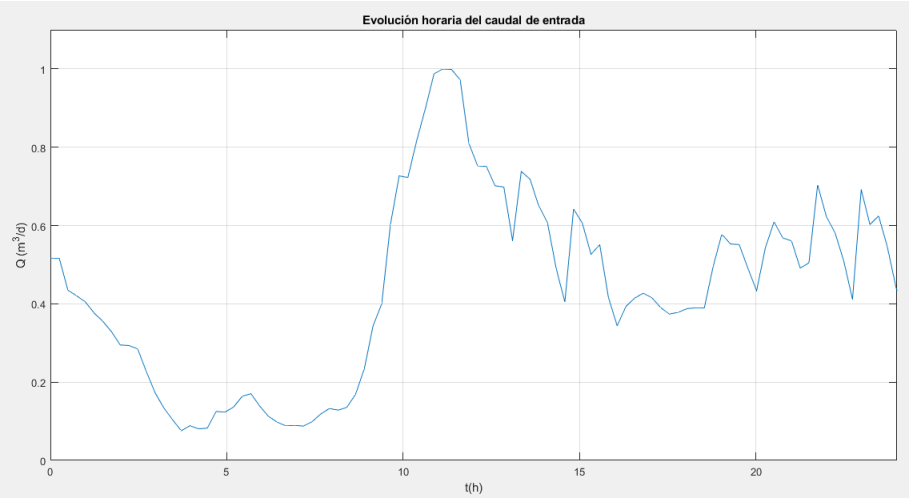


Figura 12.- Evolución intradía del caudal entrante

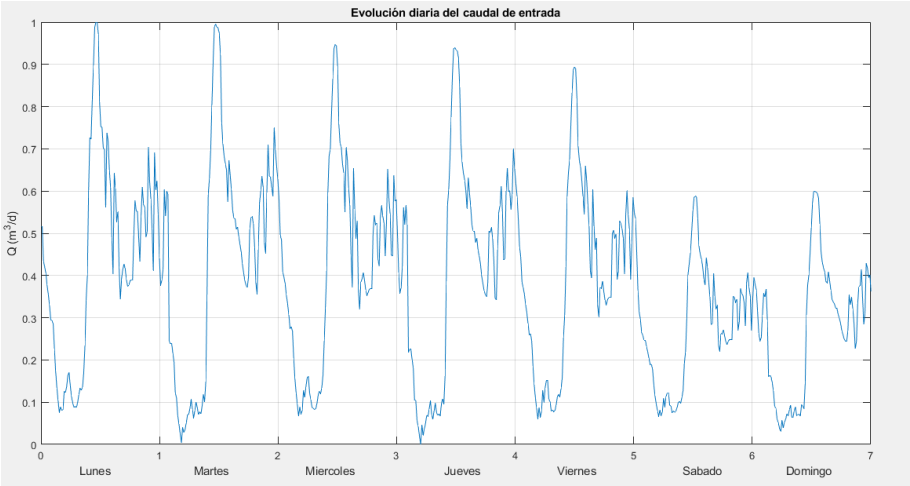


Figura 13.- Evolución intrasemana del caudal de entrada

I.6.2.- REQUISITOS DEL ALGORITMO DE CONTROL

Para que el algoritmo de control sea optimo y se pueda aplicar en una EDAR, debe cumplir los siguientes requisitos:

- El modelo de predicción debe ser capaz de simular los datos de las variables con suficiente precisión y los más eficientemente posible.
- El control debe asegurar que en ningún momento se supera el límite de amonio a la salida deseado.
- El control tiene como objetivo minimizar tanto el consumo energético como el coste económico del proceso de depuración de agua, aunque esta es una premisa secundaria por debajo del control del amonio a la salida.

I.6.3.- REQUISITOS DISEÑO APLICACIÓN

Para el diseño e implementación de la aplicación o interfaz de usuario, los requisitos más importantes y prioritarios son:

- Conseguir que el diseño sea lo más *User-friendly* posible.
- La aplicación no debe ser compleja, sino lo más intuitiva posible.
- Esta debe ser visual y que no se necesiten conocimientos profundos sobre programación o informática para instalarla y utilizarla.
- Es indispensable que sea fácilmente distribuible.
- Debe ser lo más rápida y eficaz posible.

I.6.4.- REQUISITOS LÍMITE DE AMONIO A LA SALIDA

Puesto que este es un dato que fija la ordenanza municipal de vertidos líquidos, este límite dependerá del municipio en el que se encuentre la EDAR. En concreto, en la ciudad de Castellón los límites de vertidos líquidos son los siguientes:

	Tipo A	Tipo B
Temperatura (oC)	40	40
pH	5´5-9	5´5-11
Amoniaco (mg/l)	25	40
Aldehidos (mg/l)	2	4
Aluminio (mg/l)	10	20
Arsénico (mg/l)	1	2
Bario (mg/l)	20	30
Boro (mg/l)	3	6
Cadmio (mg/l)	0´15	1

Tabla 5.- Fragmento del Artículo 9º de la ordenanza municipal de vertidos de Castellón

En el presente proyecto, el límite de amonio lo marca el BSM1, pero es un valor que se puede modificar fácilmente para que el usuario final pueda integrar la solución en cualquier tipo de estación, siempre que disponga del sensor que mida la variable a controlar.

I.7 ANÁLISIS DE SOLUCIONES

En este punto se van a estudiar las diferentes alternativas que se plantean a la hora de realizar cada una de las tareas que surgen durante el desarrollo del proyecto.

I.7.1.- SIMULACIÓN DE LA PLANTA

Puesto que no es posible excitar el sistema que se quiere controlar, ya que está en servicio y esta excitación podría tener resultados diferentes de los deseados. Se necesita diseñar un simulador sobre el que hacer las pruebas y los ajustes de los algoritmos de control.

En esta fase del proyecto se pretende obtener un programa capaz de comportarse de igual forma que el sistema real. El objetivo final de este simulador es obtener los datos de las salidas en función de los cambios a la entrada. Es decir, con el simulador se emula una situación de funcionamiento normal de una EDAR. Por tanto, una vez validado el simulador se tomarán los resultados obtenidos como equivalentes a los recogidos por los sensores instalados en un sistema real.

Para obtener datos sobre el comportamiento de una planta de depuración y además poder realizar las pruebas necesarias, es indispensable la construcción de este simulador. Para ello, es indispensable tener claros los conceptos básicos de una planta de este estilo. El proyecto actual se centra en la parte del proceso que va desde el tratamiento biológico hasta la salida de agua depurada y de lodos desechados (del punto 1 al 2-3 en la siguiente figura).

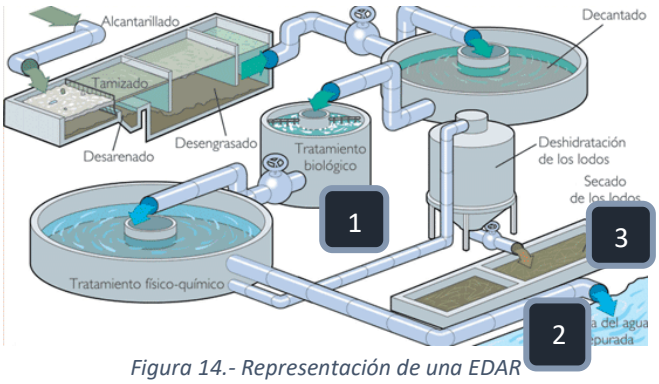


Figura 14.- Representación de una EDAR

Si se observa el sistema desde un nivel más elevado, el proceso se simplifica de una manera considerable. Aplicando esta reducción el esquema queda de la siguiente forma, la **entrada** sería el "**Agua residual**" que llega al tratamiento biológico y una **salida** que correspondería al "**Agua tratada o limpia**", además de los "**Fangos residuales**". En la siguiente imagen se ilustra la visión que se va a utilizar del sistema.

Figura 15.- Esquema entrada salida de una EDAR

Esta visión permite considerar la planta como una "**caja negra**", es decir, un proceso en el que no se conocen las variables intermedias sino únicamente las entradas y salidas. En este caso concreto el comportamiento vendrá marcado por las ecuaciones que se establecen en el **ASM1** y que son la base del ya nombrado **BSM1**. En la siguiente figura, se esquematiza la integración de las fórmulas y el diseño de la planta mediante un programa de modelado.

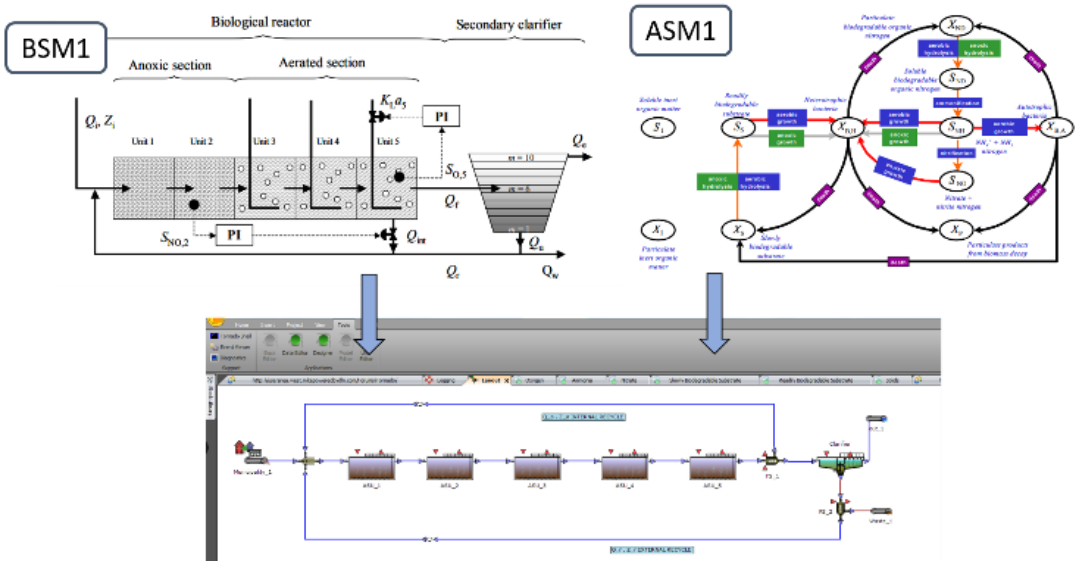


Figura 16.- Esquema de diseño de la planta en WEST

En los siguientes puntos se presentan las alternativas estudiadas, así como la selección final.

I.7.1.1.- OPCIONES DE SIMULACIÓN

Una EDAR es un sistema complejo y que depende de un gran número de variables que se rigen por una serie de ecuaciones y relaciones propias de los campos de la mecánica de fluidos y la química. Esto hace que la modelización matemática de plantas de este tipo sea una tarea realmente ardua y pesada. De hecho, la simulación mediante programas de Matlab es complejo e ineficiente, ya que limita en gran medida la posibilidad de extrapolación o modificación del sistema.

Para facilitar el proceso de modelización, diseño y cálculo de estos sistemas, existen en una serie de softwares que integran todas estas partes de una forma intuitiva y más simple. Además, al contar con apoyo gráfico, estos métodos son mucho más cómodos y permiten detectar errores fácilmente. Algunos de los programas más utilizados para este tipo de simulaciones son:

- **STOAT**: "Dynamic sewage treatment works modelling package".
- **WEST – Mike DHI**: "Modelling and simulation of wastewater treatment plants".
- **GPS-X**: "Premium Water & Wastewater Modelling and Simulation Software".

En la realización de este proyecto se ha utilizado el software **WEST** de la compañía MIKE, ya que es uno de los más potentes actualmente y, además, la universidad a través del área de mecánica de fluidos cuenta con una licencia institucional.

I.7.1.2.- WEST – TORNADO

WEST es un software que presenta grandes ventajas respecto a las formas tradicionales de cálculo, entre ellas las más destacables son:

- Permite la selección de la alternativa de proceso óptima entre las diferentes tecnologías disponibles, haciendo el diseño mucho más adaptable y personalizable.
- Es capaz de revisar/desarrollar un proyecto completo de una EDAR cualquiera.
- Acelera la puesta en marcha de plantas.
- Optimiza el proceso y minimiza los costes de operación asegurando el cumplimiento de la Normativa aplicable.

Este software lleva el análisis a otro nivel, aumentando tanto la cantidad de información como el grado de detalle de las tareas relativas al diseño o control de la explotación de una EDAR. De hecho, este es uno de sus puntos fuertes, el grado de precisión junto a la posibilidad de calibración a medida hace de este un software fiable y de calidad. Por otro lado, su interfaz de usuario cuenta con una extensa librería de componentes que se pueden utilizar para modelar prácticamente cualquier tipo de EDAR.

Todo esto hace de WEST un programa realmente potente además de *user-friendly* para la simulación y modelización tanto estática como dinámica, ya sea de una EDAR u otros tipos de sistemas relacionados con la calidad del agua. Si comparamos WEST con el resto de los softwares actuales estos serían los puntos en los que destaca:

Figura 17.- Esquema de las ventajas de WEST

Asimismo, en el proyecto, este pack destaca por encima del resto porque cuenta con el *kernell* **TORNADO**, que permite realizar la simulación mediante comandos externos a la interfaz gráfica, es decir, no se necesita lanzar las simulaciones desde dentro del propio programa. Esto hace que una vez especificado el *layout* y cada uno de los parámetros de funcionamiento del sistema, se puedan ejecutar los experimentos desde cualquier software, en el presente proyecto Matlab. Para poder realizar estos experimentos, es necesario partir de una serie de ficheros con títulos y formatos específicos que sirven de base para su ejecución.

El proceso de generación del material necesario para la realización de experimentos en Tornado es muy simple. Al guardar el diseño implementado desde la ventana gráfica de WEST, automáticamente se genera una carpeta en la que se encuentran una serie de documentos que determinan el modelo. Estos ficheros son los utilizados por el propio programa a la hora de lanzar las simulaciones y que, por lo tanto, se deben utilizar para construir el simulador [3]. En

el siguiente esquema se representa el proceso realizado para "traducir" el sistema elegido, en este proyecto (BSM1), a los ficheros que utilizará el simulador para obtener valores.

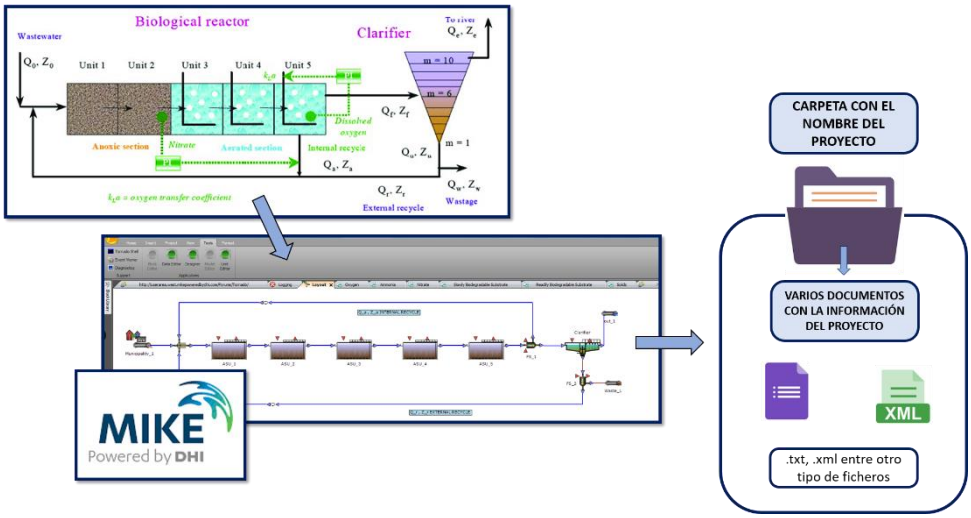


Figura 18.- Esquema de traducción de plano de EDAR a ficheros

Una vez generados los ficheros de texto y los documentos XML que definen el proyecto, se debe conocer el funcionamiento de Tornado, encargado de ejecutar las simulaciones. Tornado es una especie de motor de cálculo que ejecuta la parte matemática de las simulaciones, ya se realicen desde dentro o fuera de la aplicación.

Como se presenta en la siguiente figura, Tornado cuenta con una serie de funciones introducidas por teclado a través del símbolo del sistema, accesible desde una de las pestañas del propio WEST. En concreto, la función que ejecuta el experimento es **texec** y tiene una serie de atributos responsables de definir el comportamiento de la ejecución. Entre todas las opciones, para este simulador se utilizan -**cp** y **-i** , estos se han elegido pensando en el control de la planta, ya que **cp** permite copiar los valores finales de un experimento y definirlos como iniciales del siguiente y **-i** modifica los valores iniciales de las variables seleccionadas (en este caso las variables de entrada del sistema).

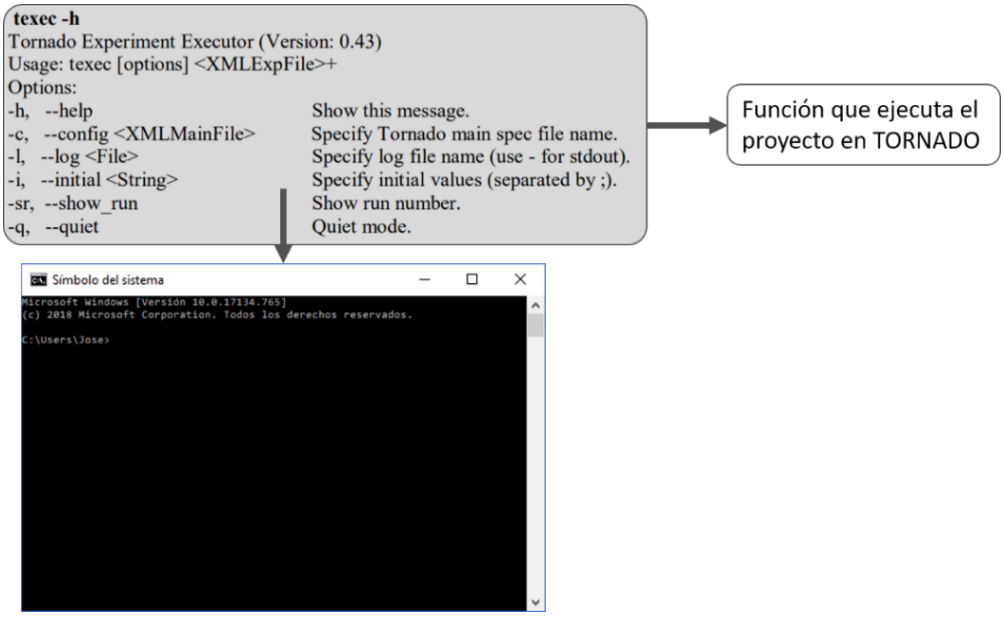


Figura 19.- Resumen funciones TORNADO y esquema de funcionamiento

Esta es una potente herramienta, pero antes de poder utilizarla de forma externa, se debe realizar una preparación previa, para evitar así el acceso a través del propio West. En la siguiente figura se presentan los pasos a seguir para realizar este proceso de preparación.

Figura 20.- Preparación del sistema para utilizar Tornado externamente

I.7.1.3.- CONEXIÓN TORNADO-MATLAB

Sobre la conexión Tornado-Matlab no existe una documentación demasiado extensa, por no decir que es prácticamente inexistente. Además, a medida que se han ido sucediendo las actualizaciones se ha ido dejando de lado la interconectividad de Tornado con otros softwares. En la siguiente figura, se presenta un esquema de las diferentes vías de conexión.

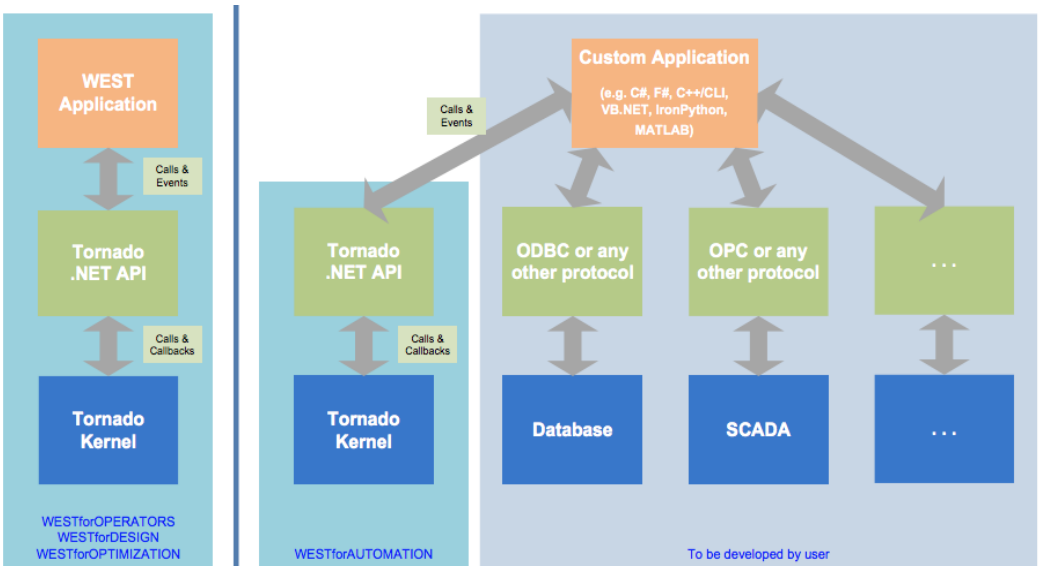


Figura 21.- Vías de comunicación con Tornado

A continuación, se muestra un resumen de las alternativas más relevantes de comunicación de Tornado con Matlab.

- **Integración de Tornado con Matlab a través de "Tornado's MEX Wrapper"**

  TornadoMEX es una biblioteca de enlace directo o DLL que actúa como MEX - librería de envoltura o *wrapper* alrededor del motor Tornado. Este tipo de archivos ***MEX*** proporcionan una interfaz entre Matlab u Octave y funciones desarrolladas en C, C++ o Fortan. De hecho, MEX significa "Matlab Executable".

  Esta opción consiste básicamente en extender el lenguaje de Matlab con una función, llamada ***TornadoMEX,*** que cuenta con 5 argumentos de entrada (modo, nombre del experimento, especificaciones sobre valor inicial, especificaciones sobre las salidas,

especificaciones sobre *loggin*) y devuelve una matriz que contiene las series temporales deseadas como columnas.

Esta opción sería una de las más recomendadas, sin embargo, existen problemas de disponibilidad y compatibilidad de la librería. A raíz de esto, se desecha la posibilidad de trabajar con esta alternativa.

- **Integración de Tornado con Matlab a través de .NET**

  Desde la versión 2009a, Matlab soporta el *framework* .NET, esto quiere decir que el código implementado en ensamblajes .NET puede ser fácilmente llamado desde el lenguaje M de Matlab. Este principio también se puede aplicar a Tornado y por lo tanto permite el uso de la .NET API de Tornado, llamada ***TornadoNET***. De este modo, esta opción se alza como la alternativa al ***TornadoMEX***, que presenta limitaciones de acceso debido a su tipo de licencia. Sin embargo, su uso viene condicionado por el pack de WEST instalado y de la versión concreta.

  Por esto y por conflictos de compatibilidad entre la versión de Matlab y WEST, no se puede utilizar esta vía y se opta por la que se presenta en el siguiente punto.

- **Llamar las líneas de comando del ejecutor de experimentos de Tornado desde Matlab**

  Puede parecer la solución menos avanzada, pero sin embargo es la más fácilmente adaptable y la que presenta menos problemas en tema de licencias e implementación. De hecho, no se necesita ningún permiso ni licencia extra para su utilización. Al tratarse de comando que se deben introducir en la consola CMD del ordenador, se utiliza el comando **system()** para poder ejecutarlo desde Matlab.

  En la figura siguiente se presenta un esquema de funcionamiento de la función ***system()***, esta cuenta con un argumento de entrada (el comando que se quiere ejecutar) y con dos salidas (el "status", que indica si el comando se ha realizado satisfactoriamente y "cmdout" que recoge el texto que se muestra por pantalla)

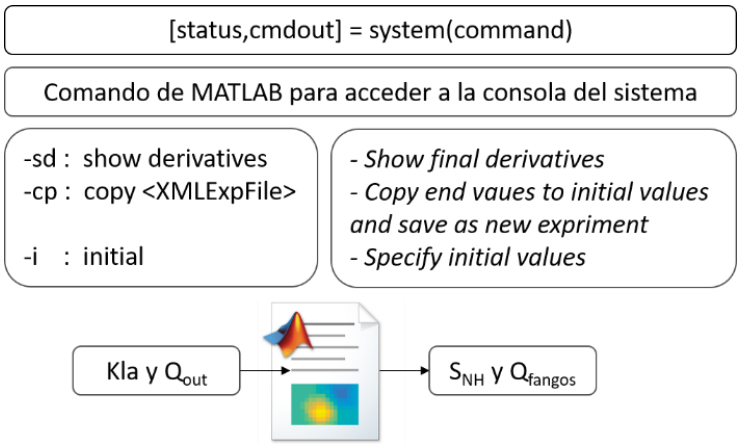


Figura 22.- Esquema de funcionamiento de system()

Para que esta alternativa funcione, se deben seguir los pasos de preparación del sistema establecidos en el apartado anterior. De esa forma se le otorgan a tanto a Matlab como al **Símbolo del sistema** los permisos y los PATHs necesarios para ejecutar los experimentos y funciones a través de Tornado.

En la realización de este proyecto se ha elegido la última opción para desarrollar el simulador de la EDAR que se presenta en el BSM1. Para facilitar la conexión, se implementan una serie de funciones que recojan las variables deseadas y calculen las salidas seleccionadas. Es decir, se va a construir una de las primeras alternativas a partir de la tercera.

I.7.1.4.- ENTRADAS Y SALIDAS DEL SISTEMA

Una vez preparado el sistema e implementada la función, el simulador es completamente ejecutable desde Matlab. Realizando la analogía con el esquema simplificado anterior (ver figura 15), el agua residual se sustituye ahora por un fichero con los datos de entrada del sistema (caudal de entrada y concentraciones), la **EDAR REAL** queda remplazada por la función que se ha implementado en Matlab y finalmente los datos de salida se recogen en una serie de variables.

Puesto que se trata de simular de la forma más fiel posible la realidad, solo se van a considerar accesibles las variables que recogen datos medibles o para las que la mayoría de las estaciones tiene instalados sensores. Una vez se da por finalizada la parte de construcción del simulador, se pasa a la verificación de este. En la siguiente figura, se muestra gráficamente la analogía del simulador y las partes del proceso real.

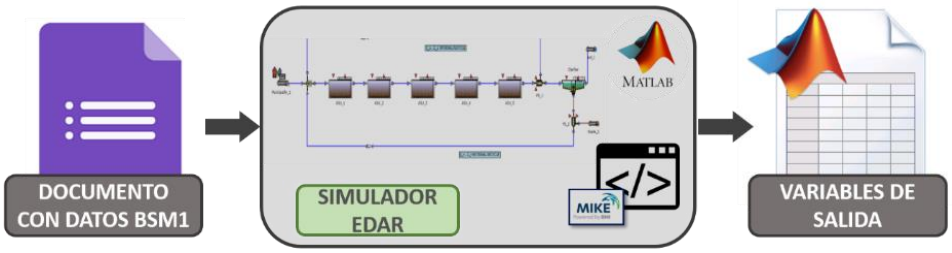


Figura 23.- Esquema de simulador en Matlab

Para trasladar el simulador al campo del control, el próximo paso es decidir cuáles son las entradas y salidas del sistema. Basando el problema en el planteamiento del BSM1, se toman como variables las presentadas en la siguiente figura.

Figura 24.- Esquema de entradas y salidas del simulador

Donde:

u_1 – señal de entrada correspondiente a K_La.
u_2 – señal de entrada correspondiente a Q de recirculación.
d – *disturbance*/perturbación – "Municipality" correspondiente a las diferentes concentraciones del caudal de entrada.
y_1 – señal de salida correspondiente a la concentración de amonio a la salida del proceso.
y_2 – señal de salida correspondiente al caudal másico de los fangos recogidos a la salida.

En el programa de modelado (WEST) estas serían las representaciones gráficas de las entradas y salidas. Por un lado, se muestra el colector de la EDAR (d – Q entrante), la bomba FS_1 que hace recircular el caudal interno (u_2 – Q_{int}) y el tanque sobre el que se va modificando la aireación (u_1 – K_La).

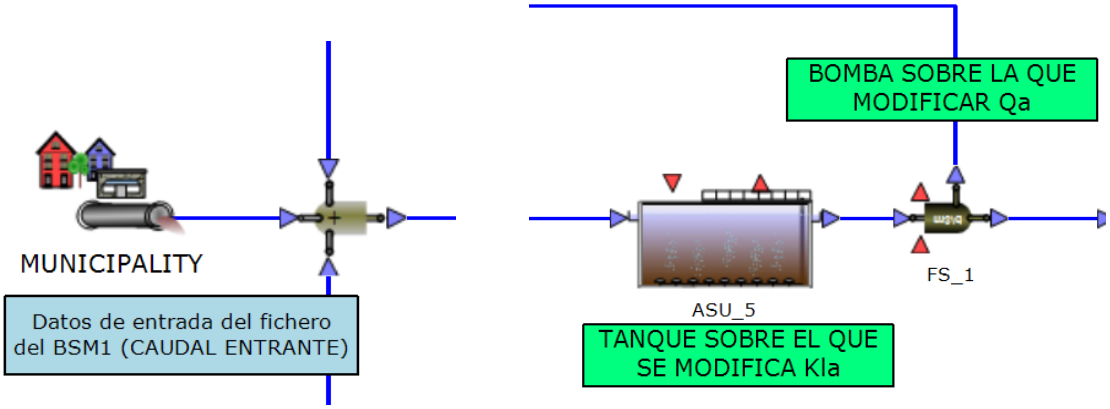


Figura 25.- Representación de las entradas de control y la perturbación en WEST

De igual modo, en la siguiente ilustración se observa la representación en WEST de los elementos que recogen las variables de salida. Por un lado, la concentración de amonio (y_1 –S_{NH}) y por el otro el caudal másico de fangos (y_2 –M_{fangos}), ambos serían el final del proceso de depuración.

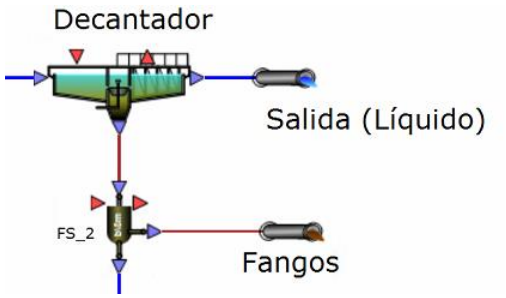


Figura 26.- Representación en WEST de los elementos de salida

I.7.2.- ESTRATEGIAS DE CONTROL

Puesto que uno de los objetivos del proyecto es la optimización y el control del sistema que se ha planteado, en este apartado se describen y analizan las estrategias y tipos considerados a la hora de desarrollar el algoritmo de control.

I.7.2.1.- SELECCIÓN DE CONTROLADOR

En esta sección se recopilan los controladores evaluados, así como el grado de adaptación al problema que se quiere resolver y sus requisitos de diseño.

CONTROL TIPO RELÉ

El **relé** o controlador *On/Off,* también conocido como Todo/Nada, realiza un control muy simple de una señal de salida. De hecho, las acciones se comportan como funciones escalón con valores prefijados en su diseño.

Existen diferentes tipos de relés en función del momento en el que se activan o desactivan las acciones de control. El **simple** se activa o desactiva cuando la variable de salida coincide exactamente con el valor de referencia, mientras que los relés con banda muerta o histéresis presentan cierto margen. En concreto, el **relé con histéresis** deja un margen de ±h respecto a la referencia, reduciendo de esta forma los cambios bruscos de la entrada. **El relé con banda muerta** es similar al anterior, pero en este caso se introduce un valor intermedio a la entrada; es decir, cuando la señal de salida está cerca de la referencia, el relé intenta no modificar su valor haciendo que la entrada influya lo menos posible. En la siguiente figura se pueden observar tanto el esquema de un relé como una comparación de las señales de los diferentes tipos de relé.

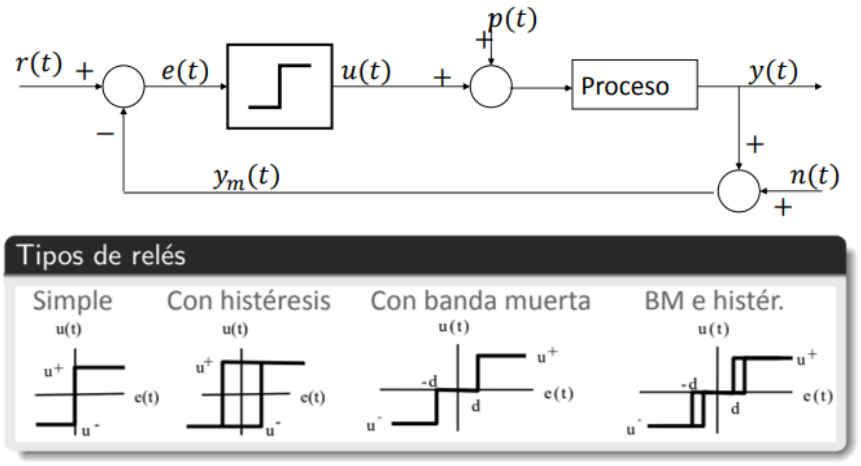


Figura 27.- Esquema control Relé

CONTROL TIPO PID

El controlador **PID** es un mecanismo de control muy popular y que está actualmente implantado en una gran cantidad de procesos industriales. Un PID tiene tres partes diferenciadas: parte **proporcional**, parte **integradora** y parte **derivativa** [10]. En la siguiente figura se muestra un diagrama de bloques diferenciando las tres fases del control.

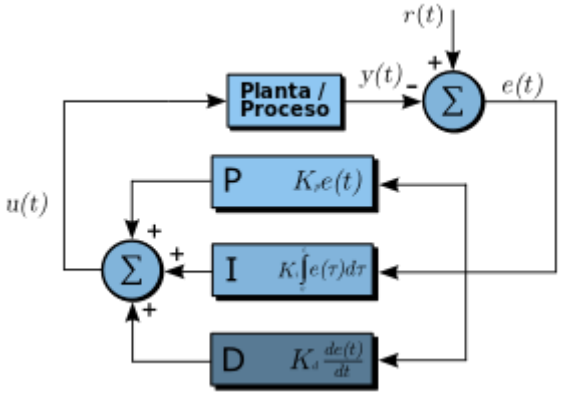


Figura 28.- Diagrama de bloques de un controlador PID

Este controlador se rige por la siguiente expresión:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)dt + K_d T_d \frac{de(t)}{dt} \qquad (1)$$

Donde:

- $\boldsymbol{u(t)}$ =Acción de control.
- $\boldsymbol{e(t)}$ = Señal de salida.
- $\boldsymbol{K_p}$ = Constante del término proporcional.
- $\boldsymbol{T_d}$ = Tiempo derivativo.
- $\boldsymbol{K_d}$ = Constante del término derivativo.
- $\boldsymbol{T_i}$ = Tiempo integrador.

Estos parámetros son los que determinan el comportamiento final del control, su velocidad, su gestión del ruido, su oscilación, etc. Aunque se trata de una estrategia simple, consigue aunar las ventajas de cada una de las acciones de control individuales que la forman.

Si bien es cierto que es una técnica interesante, no cumpliría con el requisito de optimizar el consumo, ya que no tiene en cuenta en ningún momento el comportamiento futuro del sistema.

CONTROL PREDICTIVO

El control predictivo es aquel que se basa en las salidas futuras para decidir las acciones a realizar en el presente. Esto quiere decir que, conociendo la salida en el instante de tiempo ***t,*** es capaz calcular la salida en el instante ***t+n***. A continuación, se define brevemente este tipo de control, se presenta su esquema y sus ventajas e inconvenientes.

MPC – Control predictivo basado en modelo**.** Esta estrategia se basa en el uso de un modelo para predecir la salida futura del proceso. En su funcionamiento calcula las acciones de control minimizando una función de coste en cada periodo. Una de sus características más representativas es el hecho de ser una estrategia deslizante, esto quiere decir que se aplica la primera acción de control y se vuelve a lanzar la optimización. Además, puede incorporar restricciones en las salidas o acciones de control [2]. En la siguiente figura se representa el diagrama de bloques de un MPC.

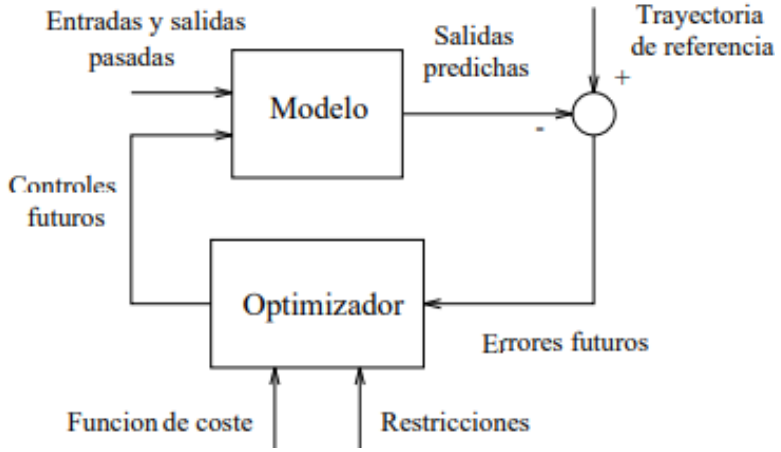


Figura 29.- Estructura básica de un MPC

Las ventajas e inconvenientes que presenta este tipo de control son las siguientes:

VENTAJAS	INCONVENIENTES
Es una estrategia bastante intuitiva.	Se necesita un modelo apropiado del proceso.
Se usa en procesos muy diversos.	La carga computacional es elevada.
Tiene una fácil extensión al caso multivariable.	
Permite el tratamiento de restricciones en entradas y salidas.	
Se aprovechan valores conocidos de referencias futuras.	

Tabla 6.- Listado de Ventajas e Inconvenientes del MPC

En el apartado, **ANEXO 2: MPC – MODEL based PREDICTIVE CONTROL**, se amplía la información sobre este control ya que es el seleccionado para llevar a cabo este proyecto.

I.7.3 MODELO DE PREDICCIÓN

En este apartado, se recogen las diferentes alternativas para llevar a cabo tanto el proceso de identificación del sistema como el de validación del modelo resultante.

I.7.3.1.- IDENTIFICACIÓN DEL SISTEMA

Para llevar a cabo el proceso de determinación del modelo, es indispensable contar con la mayor cantidad de datos del sistema excitado [1]. En este caso, se realiza una serie de procesos de excitación, cuyos resultados se pueden consultar en el apartado **I.8.2- RESULTADOS EXCITACIÓN DEL SISTEMA.**

La identificación del sistema es una de las partes más importantes del proceso de diseño del control predictivo, para llevar a cabo esta tarea, se han estudiado diferentes alternativas e incluso combinaciones entre ellas.

Uno de los puntos más importantes a la hora de analizar las alternativas es el evitar las no linealidades en las acciones de control (u_1 y u_2), ya que los optimizadores y los *solvers* no son capaces de gestionarlas. Esto quiere decir que el modelo debe ser lineal o poderse aproximar mediante un sistema lineal. Estas son las técnicas que se han estudiado:

- ***REGRESOR LINEAL***

Un regresor lineal tiene como objetivo expresar la relación entre las variables dependientes e independientes. Existen varias formas de regresores en función de las variables que utilizan para realizar la aproximación, en este caso es importante distinguir entre el regresor lineal simple y el autorregresivo. La principal diferencia entre estos dos casos es que el autorregresivo utiliza los valores de salida recién calculados para continuar calculando los siguientes. En las ecuaciones mostradas a continuación se detallan ambos tipos.

Simple:

$$y_k = \sum_{i=1}^{N} (a_i d_{k-i} + b_i u_{k-i}) \qquad (2)$$

Autorregresivo:

$$y_k = \sum_{i=1}^{N} (a_i d_{k-i} + b_i u_{k-i} + c_i y_{k-i}) \qquad (3)$$

En este caso, se ha decidido utilizar una regresión lineal autorregresiva que utiliza los valores de las variables de entrada, la perturbación medible y los valores de la salida recién calculados. En la siguiente ecuación se presenta el regresor multivariable con diferentes ordenes:

$$\begin{aligned} y_1(k) = a_0 &+ a_1 y_1(k-1) + a_2 y_1(k-2) + a_3 y_1(k-3) \dots + a_{n_{y_1}} y_1\left(k - n_{y_1}\right) \dots \\ &\dots + b_1 u_1(k-1) + b_2 u_1(k-2) + b_3 u_1(k-3) \dots + b_{n_{u_1}} u_1\left(k - n_{u_1}\right) \dots \\ &\dots + c_1 u_2(k-1) + c_2 u_2(k-2) + c_3 u_2(k-3) \dots + c_{n_{u_2}} u_2\left(k - n_{u_2}\right) \dots \\ &\dots + d_1 d(k-1) + d_2 d(k-2) + d_3 d(k-3) \dots + d_{n_{wQ}} d_1\left(k - n_{wQ}\right) \end{aligned} \qquad (4)$$

Al contar con diferentes órdenes para cada una de las variables, se puede disminuir el tamaño del regresor de una forma considerable, haciendo mucho más ligero el modelo de predicción. Si se expresa esta relación en forma matricial queda de la siguiente forma:

$$Y = X \cdot \theta \qquad (5)$$

$$X^t Y = X^t X \cdot \theta \qquad (6)$$

Donde:

- ***X*** es la matriz de variables seleccionadas para el regresor.
- ***Y*** es el vector de variables de salida.
- ***θ*** tiene la forma:

$$\theta = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n_{y_1}} \\ b_1 \\ \vdots \\ b_{n_{u_1}} \\ c_1 \\ \vdots \\ c_{n_{u_2}} \\ d_1 \\ \vdots \\ d_{wQ} \end{bmatrix}$$

Esta expresión matricial tiene más ecuaciones que incógnitas, por tanto, no tiene solución directa. Para poder obtener los valores de θ, se presentan varias alternativas:

- ***Aproximación por mínimos cuadrados***

La aproximación por mínimos cuadrados tiene como objetivo minimizar el cuadrado de la distancia entre dos curvas o conjuntos de puntos. Se expresa según la siguiente ecuación:

$$\min_{g(x_i)} \left(\left|\left| f(x_i) - g(x_i) \right|\right|_2 \right)^2 \qquad (7)$$

En Matlab, existe una forma directa de realizar la aproximación por mínimos cuadrados y se trata del comando de división por la izquierda \. Esta operación tiene diferentes efectos en función del tipo de variables que se le introduzcan, para entenderlo se va a ejemplificar con las variables *a, b* y *x* resolviendo ecuaciones del tipo *a*x=b*, donde *a* es la incógnita, en función de los diferentes tamaños. A continuación, se presentan las 3 situaciones que se pueden dar en función de la forma de *a* y *b*.

1. *a* es una matriz cuadrada → *a\b = inv(a)*b*
2. *a* es una matriz mxn y b un vector de n filas o varias columnas de n filas. → *a\b es la solución de ax=b (Calculado por eliminación gaussiana)*
3. *a* es una matriz mxn y b un vector de m filas o varias columnas de m filas. → *a\b es la solución por mínimos cuadrados del sistema de ecuaciones ax=b*

En este proyecto en concreto se da la última situación. Por lo tanto, la ecuación que define θ quedaría así:

$$\hat{\theta} = (X^t X)^{-1} X^t Y \;\rightarrow\; \theta = X \backslash Y \qquad (8)$$

Recientemente se han empezado a utilizar técnicas para verificar que, al resolver este tipo de ecuaciones, los resultados sean controlados. Por ejemplo, la de ***Regularized least squares(RLS)***. La principal ventaja es la posibilidad de introducir restricciones sobre los

valores de $\boldsymbol{\theta}$, sin aumentar en exceso la carga computacional. Está técnica se rige según la siguiente ecuación:

$$\hat{\theta} = (X^t X + \lambda I)^{-1} X^t Y \qquad (9)$$

- ***Optimizador + Solver: YALMIP + solver compatible.***

Otra opción para resolver el sistema anterior es la utilización de optimizadores y *solvers*. Ambas herramientas de Matlab de una gran potencia y de gran utilidad para el tipo de proyecto que se está realizando.

En este caso concreto, se ha elegido el optimizador YALMIP. En el apartado "***I.7.4.1-YALMIP***" se explica el funcionamiento y la estructura de este optimizador, así como los *solvers* compatibles que se han barajado como posibles alternativas.

En el proceso de identificación, las restricciones van enfocadas principalmente a evitar los siguientes puntos negativos de la aproximación por mínimos cuadrados:

- **CONTROL SOBRE LOS COEFICIENTES.** El optimizador permite controlar el valor de los coeficientes manteniéndolos dentro del rango deseado.
- **PRIORIDADES DE LA IDENTIFICACIÓN.** Posibilita asignar prioridad a una parte concreta de la identificación.

- ***WIENNER-HAMMIERSTEIN***

Esta estrategia consiste en transformar un modelo no lineal, en uno lineal con no linealidades externas a la entrada y salida. Esto hace que se pueda trabajar con optimizadores que eviten las no linealidades, siguiendo la estructura representada a continuación.

Figura 30.- Esquema de sistema Wienner-Hammierstein

Este tipo de modelo matemático es muy útil en análisis y simulación de sistemas, en predicción de series temporales, interpolaciones, supresión de ruido, detección de fallos, etc [4]. De hecho, como los sistemas reales son no lineales por naturaleza, el modelado e identificación de estos sistemas es una fase clave.

En Matlab, existe una *ToolBox* que permite la identificación de modelos no lineales, permitiendo además la utilización del teorema de WIENNER-HAMMIERSTEIN. Seguidamente, se explica el funcionamiento de esta herramienta y cuál es el proceso de extracción de resultados.

- ***SYSTEM IDENTIFICATION - Matlab***

Dentro de Matlab, existe una toolbox especifica para la identificación de modelos. Esta es realmente útil ya que permite a la vez modificar el modelo, comparar las variantes, así como filtrar o modificar las entradas. En la siguiente figura se muestra un esquema del funcionamiento de la misma. Además, en todos los pasos del proceso de

identificación esta herramienta permite representar gráficamente tanto las entradas como las salidas o los nuevos modelos.

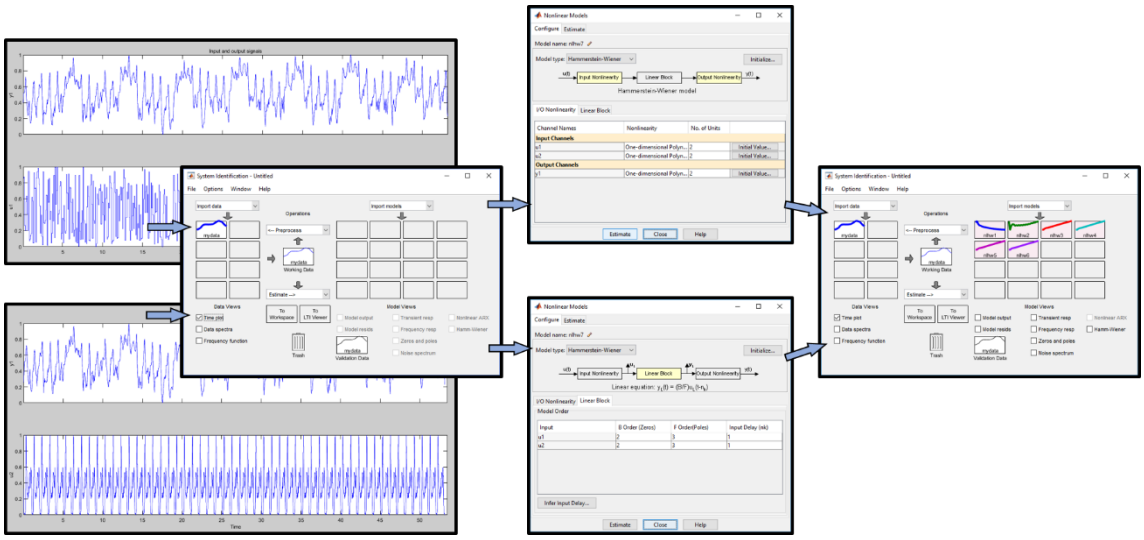


Figura 31.- Esquema de funcionamiento de la herramienta de identificación de Matlab

Una vez seleccionado el tipo de modelo que se quiere identificar, en este caso Wienner-Hammierstein, se seleccionan las no linealidades y el modelo lineal, se lanzan las iteraciones internas y se obtienen modelos al con su grado de ajuste y validez. A continuación, se presenta un ejemplo de representación y comparación de algunos modelos obtenidos.

Figura 32.- Resumen valores de salida de la herramienta de identificación

- ***VOLTERRA***

Los modelos de Volterra representan la ampliación de los modelos de respuesta ante impulso y vienen definidos en términos generales por:

$$y(k) = h_0 + \sum_{n=0}^{\infty} \sum_{n=0}^{\infty} \overset{n}{\dots} \sum_{n=0}^{\infty} h_n(i_1, \dots, i_n) \cdot \dots u(k - i_1) \dots u(k - i_n) \qquad (10)$$

Este tipo de modelo normalmente presenta un buen comportamiento y su estructura es aprovechable en el diseño de controladores, especialmente en modelos de segundo orden [8][5]. Su expresión en sistemas de memoria acotada y de segundo orden queda así:

$$y(k) = h_0 + \sum_{i=1}^{N_1} h_1(i) \cdot u(k-i) + \sum_{i=1}^{N_2} \sum_{j=1}^{N_2} h_2(i,j) \cdot u(k-i)u(k-j) \qquad (11)$$

Este método se desecha ya que a la hora de implementar el MPC los optimizadores no serían capaces de gestionar las no linealidades que este introduciría al sistema.

Para este proyecto se utiliza la técnica de **regresión lineal** combinada con el modelo de **Wienner-Hammierstein,** de este modo se consigue un sistema en el que las acciones de control no se ven afectadas por ninguna no linealidad.

I.7.3.2.- VALIDACIÓN DEL MODELO DE PREDICCIÓN

Para que un modelo se considere como apto, se debe validar lanzando el modelo de predicción para unos valores diferentes a los que se utilizaron para generarlo. Es decir, primero se cargan los archivos de uno de los experimentos realizados, a través de los cuales se extrae el regresor. Una vez construido este modelo, se cargan los datos de otro de los experimentos y se compara el valor obtenido desde el estimador con el valor real de la variable que se está estimando.

Por lo tanto, la estructura de la tarea de identificación y validación del modelo es la siguientes:

1. Excitación del sistema.
2. Tratamiento y preparación de los datos de la excitación.
3. Estudio de las diferentes alternativas para la construcción del modelo.
4. Construcción del modelo que se considere más adaptado.
5. Excitación del sistema con unas condiciones diferentes.
6. Aplicación del modelo a los nuevos datos.
7. Comparación y análisis.
8. Validación.

I.7.3.3.- APLICACIÓN PARA IDENTIFICACIÓN DEL MODELO

Puesto que esta tarea se ha de repetir para cada sistema sobre el que se quiera aplicar un control predictivo (basado en modelos), de entre todas las opciones disponibles se ha decidido diseñar una pequeña interfaz que facilite el proceso. Esta aplicación tiene como principal objetivo simplificar el proceso de identificación del sistema pensando en el usuario final.

En ella, se pueden ir probando diferentes ordenes, *delays,* grados y "filtros de no linealidades", hasta hallar la configuración óptima. Para ello se seleccionan unos datos de partida, a partir de los cuales se calcula el modelo. De igual modo se seleccionan los datos de validación del modelo y se representan en la gráfica inferior, a fin de comprobar visualmente que la precisión de la identificación es independiente de los datos de partida. En la siguiente figura se muestra el aspecto de esta aplicación, cuyo funcionamiento se detalla en el punto ***III.7.- MANUAL DE APLICACIÓN DE IDENTIFICACIÓN.***

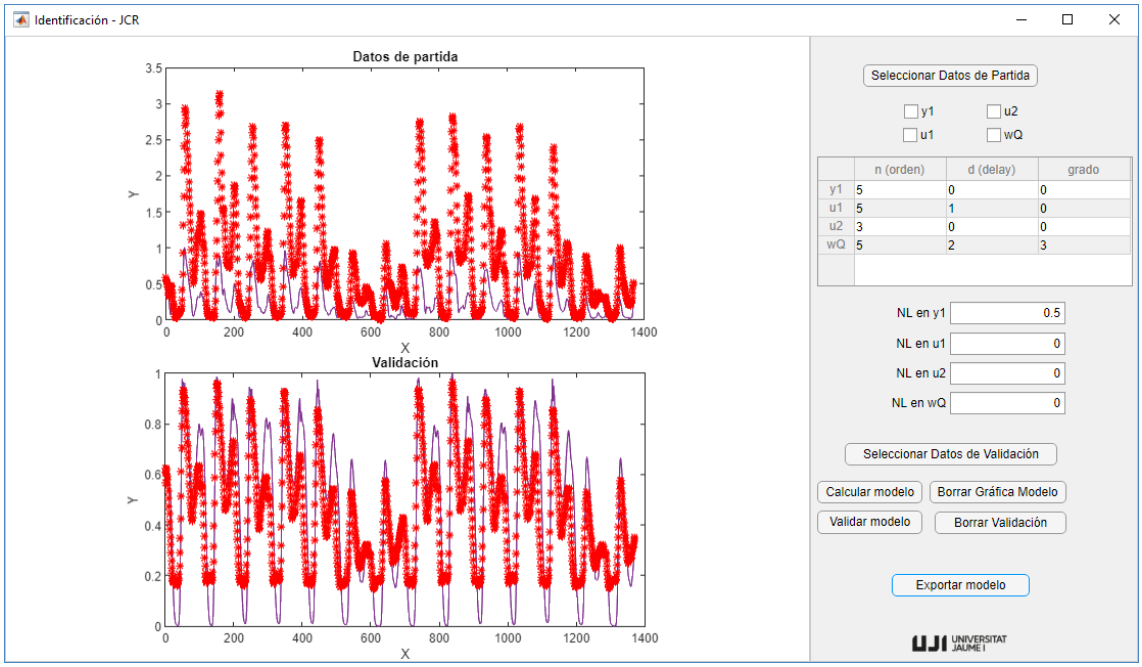


Figura 33.- Captura de aplicación para identificación del modelo

I.7.4 OPTIMIZADORES

Se denomina optimizador a aquel programa o conjunto de programas que selecciona entre una serie de variables de decisión los valores que minimizan una función objetivo.

Hoy en día, las inecuaciones lineales de matriz (LMI) y sus técnicas se han alzado como una herramienta de diseño de gran alcance y calidad en áreas que van desde la ingeniería de control hasta el diseño estructuran, pasando por la identificación de sistemas. Existen varias ventajas y factores que hacen esta estrategia realmente atractiva, entre ellos se encuentran los siguientes:

- Una gran cantidad de especificaciones de diseño y restricciones pueden expresarse en forma de LMI.
- Una vez se haya formulado correctamente en LMI, un problema o caso concreto puede ser resuelto de forma exacta mediante optimización convexa eficiente.
- Gracias al LMI, se pueden resolver problemas o sistemas con múltiples limitaciones u objetivos que carecen de solución analítica. Es decir, esta nueva metodología se erige como alternativa a los clásicos métodos analíticos.

Existen varios optimizadores que funcionan con este tipo de ecuaciones, entre ellos se han analizado los siguientes:

- **YALMIP.**
- **CVX.**

I.7.4.1.- YALMIP

Este es un paquete de programas y funciones que se descarga para que Matlab pueda ejecutarlas. El nombre de este optimizador significa "***Yet Another LMI Parser***", donde LMI hace referencia a *Linear Matrix Inequalities*. Este tipo de optimizador utiliza *Solvers* para obtener los valores de optimización.

Este es un optimizador libre que desarrolló Johan Löfberg y que va actualizando anualmente. La estructura de esta herramienta es la siguiente:

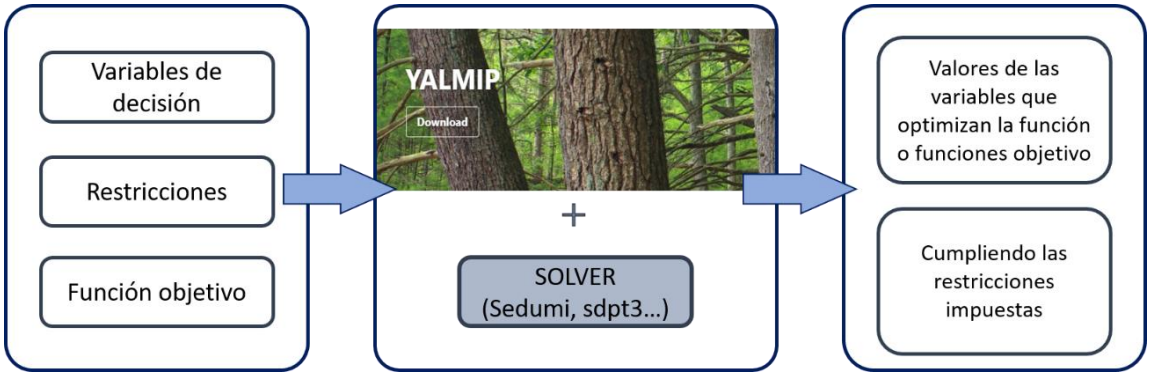


Figura 34.- Esquema resumen de funcionamiento de YALMIP

Además, al contar con un entorno de modelización, **YALMIP** permite modificar tanto las condiciones y características de la optimización como el *solver* que se quiere utilizar. En la siguiente figura se presenta el flujo de información y tareas, desde el Usuario hasta el *solver*.

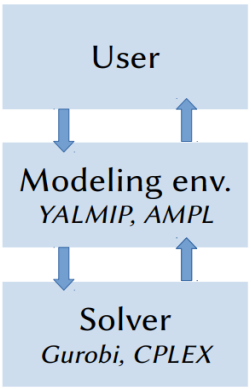


Figura 35.- Esquema de intercambio de información en optimizadores

En la siguiente tabla se recogen algunos términos y funciones relacionados con el optimizador, cuyo significado se considera necesario para comprender los apartados futuros:

FUNCIÓN	DESCRIPCIÓN
sdpvar()	Se utiliza para definir "Variables simbólicas de decisión"
sdpsettings()	Sirve para comunicar opciones a YALMIP y a los *Solvers*
optimize()	Es el comando utilizado para resolver los problemas de optimización.
value()	Se usa para extraer el valor numérico de una variable de decisión.
repmat()	Función de MATLAB que sirve para replicar un número en forma de Matriz o Array.
yalmiptest	Ejecuta una serie de ejemplos para probar la instalación.

Tabla 7.- Listado de función para YALMIP

I.7.4.2.- CVX

Otra opción para el modelado de sistemas convexos de optimización basada en Matlab es CVX. Este es un sistema de modelado para la construcción y resolución de ***disciplined convex programming*** (DCP). Soporta una gran variedad de tipos de problemas entre los que se encuentran los programas lineales y cuadrático (LP/QP), cónicos de segundo orden y programas semidefinidos (SDP) que son los que nos interesan en este proyecto. Cuenta con bastantes *solvers* lo que la hace una herramienta a tener en cuenta.

Figura 36.- Logo de CVX

Este optimizador es muy similar al presentado en el punto anterior, permite tener en cuenta tanto restricciones y objetivos expresadas usando estándar Matlab.

I.7.4.3.- MATRICES PARA EL OPTIMIZADOR

Para que el proceso de optimización sea eficiente y se realice de forma dinámica y lo más rápidamente posible. Estas matrices sirven para ir almacenando el estado de las variables de decisión y que se vaya actualizando su valor. Para ello se utiliza una ecuación matricial que presenta la siguiente formula:

$$x_{k+1} = A * x_k + B * u_k + Q * wQ_k + C \qquad (12)$$

Donde los vectores corresponden a:

$$x_k = \begin{bmatrix} y_{k+1} \\ y_k \\ y_{k-1} \\ \vdots \\ y_{k-n_y+1} \\ u_{1_k} \\ u_{1_{k-1}} \\ \vdots \\ u_{1_{k-n_{u_1}+1}} \\ u_{2_k} \\ \vdots \\ u_{2_{k-n_{u_2}+1}} \end{bmatrix} \quad x_{k-1} = \begin{bmatrix} y_k \\ y_{k-1} \\ y_{k-2} \\ \vdots \\ y_{k-n_y} \\ u_{1_{k-1}} \\ u_{1_{k-2}} \\ \vdots \\ u_{1_{k-n_{u_1}}} \\ u_{2_{k-1}} \\ \vdots \\ u_{2_{k-n_{u_2}}} \end{bmatrix} \quad u_k = \begin{bmatrix} u_{1_k} \\ u_{2_k} \end{bmatrix} \quad wQ = \begin{bmatrix} wQ_k \\ wQ_{k-1} \\ \vdots \\ wQ_{k-n_q} \end{bmatrix}$$

Y las matrices corresponden a:

$$A = \begin{bmatrix} a_1 & a_2 & \cdots & b_2 & b_3 & \cdots & b_n & c_2 & c_3 & \cdots & c_n \\ 1 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \ddots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & 0 & \ddots & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} b_1 & c_1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \quad Q = \begin{bmatrix} d_1 & d_2 & \dots & d_n \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad C = \begin{bmatrix} a_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Para entender de forma clara el funcionamiento de estas matrices, en la siguiente figura se explica como la relación entre un paso del vector de estados y el siguiente mantiene una relación diagonal, salvo el último valor de cada una de las variables (ya que no es representativo para el siguiente paso). Además, se observa que las variables de decisión provienen de las otras matrices.

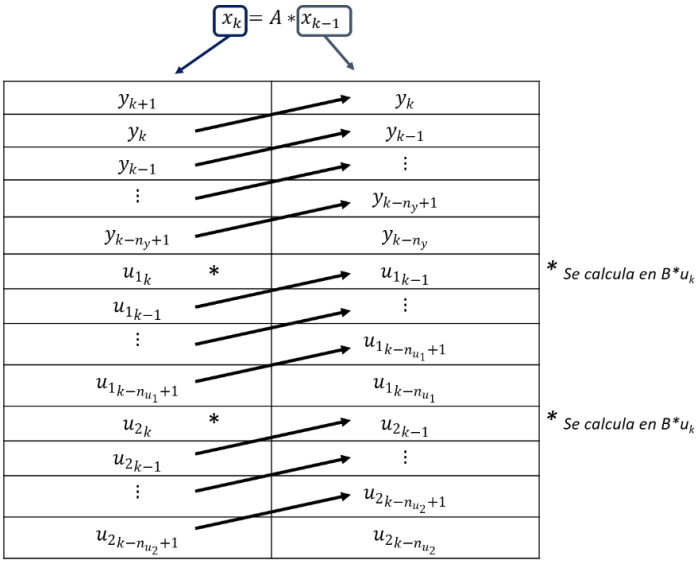


Figura 37.- Explicación del avance de X_k en cada iteración

De esta forma, se pueden introducir y actualizar las variables del predictor en el algoritmo de control. En el **ANEXO 5: CÓDIGO UTILIZADO** se el programa que construye las matrices.

I.7.4.4.- RESTRICCIONES

Las restricciones en los casos de optimización son los requisitos o premisas que este debe intentar alcanzar. Gracias a estas, se puede acotar el problema, a priori con infinitas soluciones. En el caso de YALMIP las restricciones se introducen de la siguiente manera, (el siguiente extracto es un ejemplo de introducción simple):

```
% Define variables
x = sdpvar(10,1);
% Define constraints
    Constraints = [sum(x) <= 10, x(1) == 0, 0.5 <= x(2) <= 1.5];
% Define an objective
Objective = x'*x+norm(x,1);

    % Solve the problem
    sol = optimize( Constraints, Objective, options);
```

También existe la posibilidad de insertar o definir restricciones más complejas e incluso que vayan modificándose a medida que avanza el experimento. Lo más común será utilizar bucles y bloques condicionales para implementarlas.

En este proyecto las restricciones que se han tenido en cuenta son las siguientes:

- Las acciones de control deben estar dentro del rango de funcionamiento. Para verificar esto se añade:
  `0<=((u{k}(1)))<=1` `0<=((u{k}(2)))<=1`
- El nivel de amonio debe estar siempre por debajo de los límites legales fijados en ***refvar*** (variable a definir). Y esto se expresa de la siguiente manera:
  `x(1)<=refvar`
- El cambio entre acciones de control no debe ser brusco, es decir, que la diferencia entre una acción de control y la aplicada justo antes no supere el 40% del valor de la acción. Al estar normalizadas las variables, esta restricción tiene esta forma:
  `-0.4<=(u{k}-u{k-1})<=0.4`

I.7.4.5.- FUNCIÓN OBJETIVO

La función objetivo es aquella función que el optimizador intenta minimizar, respetando en todo momento las restricciones que se le impongan. De hecho, se llama ***Objective*** porque como su propio nombre indica es el fin u "objetivo" a alcanzar cuando se lanza el optimizador.

En el caso concreto en el que se encuadra el proyecto, se pretende minimizar la función de coste eléctrico del proceso de depuración de aguas residuales. La siguiente expresión corresponde a la función de coste debido a la acción de control u_1 (K_La) que, al activarse mediante un soplador, conlleva un consumo directamente proporcional al valor de dicha variable y los diferentes periodos tarifarios entre t y HT (Horizonte Temporal [h]).

$$J_{consumo\ eléctrico} = precio_{(t=1\ am)} * \left(u_{1\,t=00:15\ am} + \cdots + u_{1\,t=01:00\ am}\right) + \cdots$$
$$+ precio_{(t=HT\ am)} * \left(u_{1\,t=(HT-1):15\ am} + \cdots + u_{1\,t=HT:00\ am}\right) =$$
$$= \sum_{i=t}^{t+HT} precio_{(t=i)} * \left(u_{1\,t=(i-1):15\ am} + \cdots + u_{1\,t=i:00\ am}\right) \qquad (13)$$

Adicionalmente esta función objetivo se puede modificar para tener en cuenta el coste derivado de la acción de control (u_2) quedando de la siguiente manera.

$$J_{consumo\ eléctrico_2} = \sum_{i=t}^{HT} precio_{(t=i)} * \left(u_{1\,t=i} + costnate_{peso} u_{2\,t=i}\right) \qquad (14)$$

Donde $costnate_{peso}$ es un parámetro que determina la importancia en términos económicos de esta acción de control.

Por otro lado, existe una segunda función objetivo que pretende minimizar el fallo en el seguimiento de la referencia. Es decir, una función que asegura que los vertidos están controlados (y_1 < límite legal) y que solamente se activa en caso de que la optimización del controlador principal no sea factible. Esta se define según la siguiente expresión:

$$J_{error\ de\ seguimiento} = \left(y_{1\,k} - límite\ y_1\right)^2 \qquad (15)$$

I.7.4.6.- SOLVERS

Los optimizadores en general y YALMIP en concreto, cuentan con una serie de *solvers*, que como su traducción directa indica, son "Solucionadores" esto quiere decir que son los métodos o algoritmos para resolver el caso a optimizar. A continuación, se presentan cuatro de los más utilizados y que se han considerado en la realización de este proyecto.

SEDUMI

SeDuMi tiene como objetivo resolver problemas de optimización convexa relacionados con ecuaciones lineales e inecuaciones, *second-order connne constraints* y restricciones semidefinidas (LMI). Se trata de un proyecto *Open-source* y como tal, sus beneficios vienen de contribuciones de los usuarios o distribuidores. En concreto, los máximos contribuyentes a este proyecto son los desarrolladores de YALMIP y CVX, dos de los *frameworks* para optimización que usan SeDuMi como *solver*.

La última versión de este *solver* es fácilmente descargable desde la página web de los autores, que además ponen a disposición una serie de Manuales y ejemplos de utilización. Las

últimas versiones se distribuyen bajo la licencia publica general o GNU *General Public License* 2.0, un tipo de licencia de derecho de autor ampliamente utilizado entre los distribuidores de software libre y código abierto.

MOSEK

Mosek es otro de los *solvers* que se han tenido en cuenta, ya que está disponible de forma gratuita para fines académicos. ***Mosek Optimization Suite*** es un paquete de software que permite resolver problemas de grandes dimensiones con muchas restricciones y/o variables. Además, Mosek provee interfaces para los principales lenguajes de programación como C, Java, Matlab, .NET, Python y R.

Este paquete permite la resolución eficiente de la mayoría de las clases generales de problemas cónico. Algunas de las restricciones y expresiones que pueden ser modeladas usando la forma cónica son las siguientes:

- Optimización cuadrática cónica.
- Optimización potencial cónica.
- Exponencial cónica.
- Optimización Semidefinida.

PENBMI

PENBMI tiene como principal objetivo solucionar problemas de optimización con objetivo cuadrático y restricciones bilineales inecuaciones de matrices. Está dirigido tanto a la resolución de LMI y BMI tanto densos como dispersos o de pequeña o gran escala.

Algunos de los puntos fuertes de este software son:

- Su tratamiento eficiente de cualquier tipo de dispersión en patrones en datos de problemas.
- Trabaja con funciones objetivo cuadráticas.
- Cálculo de problemas de viabilidad de BMI y LMI.
- Va dirigido a problemas a gran escala.

Esta es una herramienta verdaderamente potente que además puede utilizarse a través del optimizador usado en este proyecto, ***YALMIP***.

SDPT3

Este software está diseñado, como los anteriores, para solucionar problemas de programación cónica cuyas restricciones de cono son producto de conos semidefinidos, conos de segundo orden, ortantes no negativas y espacio Euclidiano y cuya función objetivo es la suma de funciones lineales.

Este *solver* está programado principalmente para su uso en Matlab, aunque existe la versión compilada en C. Esta herramienta es algo menos potente que las anteriores. Pero se evaluará de igual forma.

I.7.4.7.- SELECCIÓN OPTIMIZADOR Y SOLVERS

Una vez presentada la información sobre los optimizadores y *solvers,* se ha decidido que el optimizador seleccionado es YALMIP, ya que presenta ventajas notorias frente al resto de opciones.

En cuanto a la selección de los *solvers* se han realizado una serie de experimentos y test de rendimiento que se recogen en el apartado **I.8.5.- RESULTADOS SOBRE EL CONTROL DEL SISTEMA**. Además, se han detectado problemas por la elevada carga computacional a la que se ve sometida el optimizador, en el siguiente apartado se proponen métodos para aligerar este proceso.

I.7.5 MÉTODOS PARA ALIGERAR EL OPTIMIZADOR

Los problemas de optimización suponen una elevada carga computacional y por lo tanto ralentizan el proceso de control, para evitar este fenómeno se proponen una serie de métodos:

- Ser **más permisivo con los valores que se encuentran más alejados** del punto de partida. De esta forma, cuando el horizonte de predicción es grande, se va reduciendo el grado de exigencia. Permitiendo que las predicciones superen los valores límite. Este método nace tras observar que los modelos son menos exactos con el paso del tiempo, prediciendo mucho mejor a corto que a largo plazo. El único cambio que introduce esta opción es la incorporación de la siguiente restricción:

$$x(1) - refvar * \left(1 + \frac{alpha * k}{Horizonte\ en\ pasos}\right) \leq 0 \qquad (16)$$

  Donde:

  - ***x*** es el vector de estados.
  - ***refvar*** es la variable interna del optimizador que se sustituye por el valor límite de la salida.
  - ***alpha*** es la constante a determinar que influye en el grado de permisividad a futuro.
  - ***k*** es la variable del bucle que hace avanzar el programa de simulación con MPC.
  - ***Horizonte en pasos*** es el parámetro del control que define el número de muestras que se predicen en cada vuelta del bucle.

En la siguiente gráfica se presenta el comportamiento del límite aplicando este crecimiento a partir de la muestra central del horizonte de control.

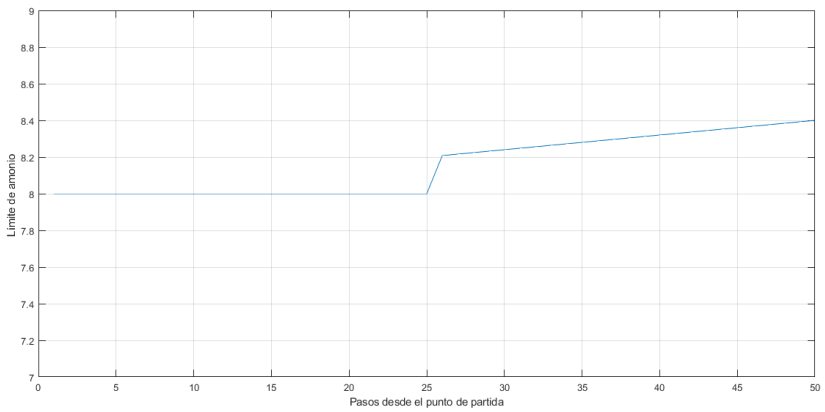


Figura 38.- Representación del avance del valor límite

- **Modificar el periodo de muestreo** fijando las acciones de control a medida que el modelo avanza hacia el futuro. Esto se traduce en una reducción de las acciones a calcular y en un aumento de la velocidad de cálculo. En la siguiente gráfica, se puede observar el comportamiento de las acciones de control si se utiliza este método.

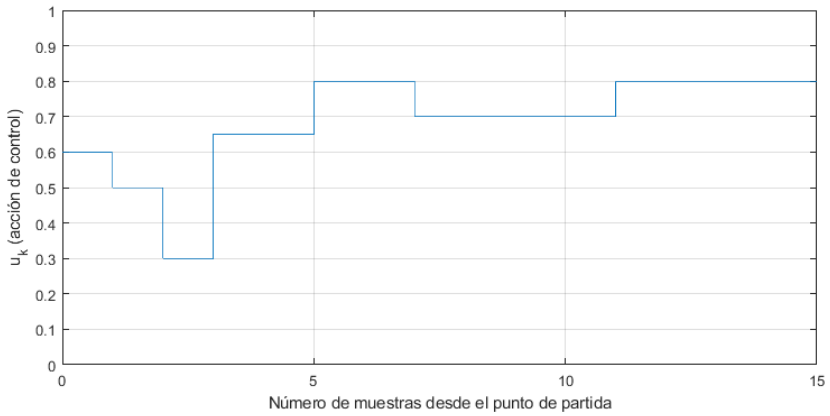


Figura 39.- Representación de la modificación del periodo de cálculo de acciones

Al tratarse de dos métodos de cosecha propia, se van a realizar pruebas con ambos, para descubrir el óptimo. En apartado de resultados, se recopila la información sobre los test ejecutados.

I.7.6 INTERFAZ DE USUARIO

I.7.6.1.- SELECCIÓN TIPO DE INTERFAZ

La interfaz gráfica de usuario o GUI (*Graphical User Interface)* es una interfaz de usuario que incluye elementos gráficos como ventanas, iconos y botoneras. El termino fue acuñado en 1970 para diferenciar este tipo de interfaz a las de puramente basadas en texto o TUI (*Textual User Interface*), como las interfaces de comando de líneas. Si bien es cierto que hoy en día prácticamente todas las interfaces son gráficas [7].

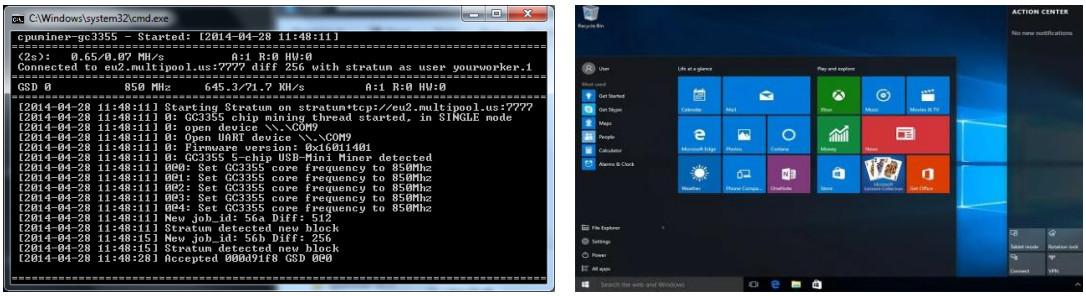


Figura 40.- Interfaz basada en texto (izq.) / Interfaz gráfica (dcha.)

En la figura anterior, se puede observar la comparación entre una interfaz de texto y otra gráfica. La principal diferencia es la comodidad de cara al usuario, ya que en el caso de las TUI el usuario debe ser mucho más especializado en la aplicación, mientras que la interfaz gráfica es mucho más intuitiva para el usuario.

Este tipo de interfaces son en parte las precursoras del avance tecnológico en ciertas áreas donde el usuario no siempre tiene un nivel de conocimientos en informática elevados. Por ejemplo, los ordenadores han podido llegar a todos los hogares gracias a la interfaz fácil y agradable de la que disponen. Además, en aplicaciones industriales, para realizar los controles

en directo es vital que la interfaz sea lo más clara y sencilla posible para que no sea posible ningún tipo de malentendido.

En este proyecto se ha seleccionado una interfaz de este tipo para facilitar al máximo la comunicación entre el usuario y el sistema. Desde esta se podrá controlar las características principales del control que se va a implementar, así como se monitorizarán las variables de salida.

I.7.6.2.- OPCIONES PARA REALIZAR INTERFACES EN MATLAB

Matlab es una herramienta muy potente y que presenta una gran cantidad de funcionalidades, además es uno de los softwares de cálculo más utilizado a nivel de investigación. Sin embargo, en la batalla del desarrollo de aplicaciones de usuario nunca ha sido una de las principales alternativas. A pesar de ello, se decide utilizar este software para la realización de la aplicación ya que facilita la integración de las soluciones de control adoptadas a lo largo del proyecto.

A la hora de realizar una interfaz de usuario mediante Matlab, existen tres métodos:

1. **Funciones de Matlab**
2. **GUIDE**
3. **App Designer**

A continuación, se va a analizar la validez de cada uno de los métodos anteriores, mostrando sus ventajas y desventajas con el fin de seleccionar la solución más adecuada para el proyecto actual.

FUNCIONES DE MATLAB

Una de las más conocidas y utilizadas técnicas de interacción entre código y usuario son las funciones. Estas se utilizan para transformar una serie de inputs en las salidas deseadas. Pese a ser de gran utilidad para la mayoría de los casos y usos de Matlab, requieren de una serie de conocimientos en el tema, además de lo poco intuitivas que resultan. De hecho, no llegarían a ser interfaces de usuario como tal.

Estas funciones se suelen utilizar para el cálculo de procesos estandarizados o repetitivos, es decir que se pueden reutilizar entre proyectos o programas. Su definición es muy simple y rápida desde el editor de Matlab. En la siguiente imagen se muestra un ejemplo de la definición de una función utilizada en el proyecto.

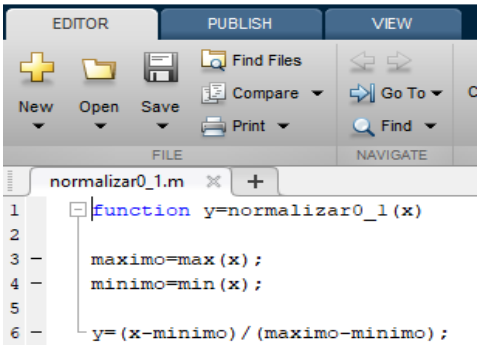


Figura 41.- Ejemplo de definición de función

GUIDE

Desde las primeras versiones de Matlab, el programa cuenta con el editor de interfaces de usuario llamado GUIDE. Este editor permite diseñar una GUI de una forma no muy compleja, aunque bastante limitada. La principal desventaja de este tipo de interfaces es la reducida galería de componentes con la que cuenta. En la figura que se presenta a continuación se puede observar que el editor cuenta con algunos botones, campos editables, ejes para gráficas, etc.

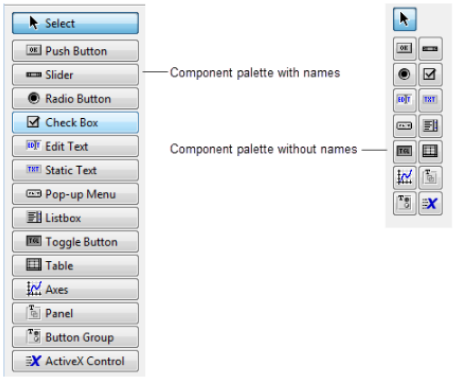


Figura 42.- Galería de componentes de GUIDE

Para definir este tipo de interfaces, se utilizan en paralelo el editor de la figura y el editor de código que se genera automáticamente. Mientras que el primero permite añadir y colocar los diferentes componentes disponibles en la galería, el segundo genera código para cada uno de los elementos y les asigna funcionalidades. Uno de los principales puntos a tener en cuenta, a la hora de programar interfaces mediante este método, es la variable de ***handless***. Está va asociada a cada interfaz y permite compartir información entre las funciones de diferentes elementos. En la siguiente figura, se pueden observar en paralelo, la ventana de ***front-end*** o parte visible de la aplicación y la de ***back-end*** o procesado de entradas.

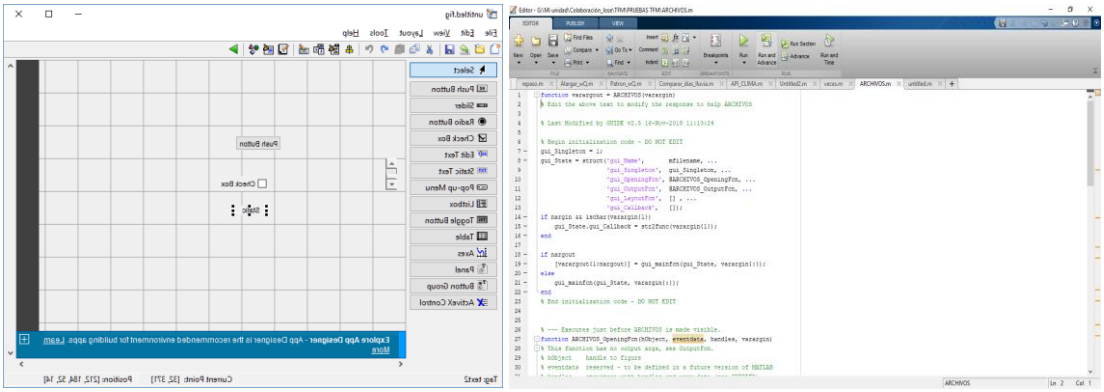


Figura 43.- Ventana de editor de figura (izq.) Editor de código asociado (dcha.)

La principal ventaja de esta herramienta es que se trata de una tecnología muy madura, de la que existen gran cantidad de ejemplos y documentación. Esto hace que el proceso de diseño sea mucho más simple y cómodo. Sin embargo, las aplicaciones realizadas con esta herramienta resultan difíciles de personalizar y por tanto acaban siendo prácticamente todas de un estilo muy similar.

APP DESIGNER

Desde la versión 2019a, Matlab cuenta con la herramienta "App Designer". Esta es la primera versión de una aplicación de diseño de aplicaciones mucho más *User friendly* y con un grado de personalización mucho más elevado. A continuación, se muestran las diferentes pestañas o secciones que presenta un proyecto de este tipo.

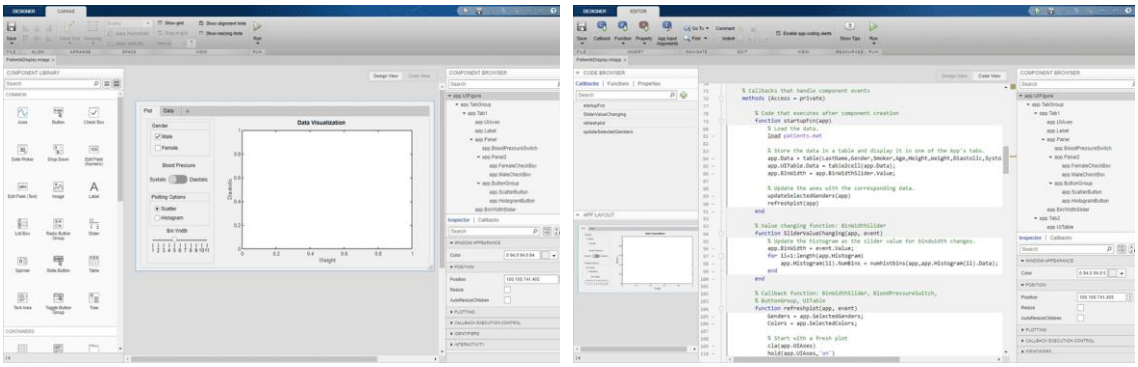


Figura 44.- Ventana con los dos modos de programación de App Designer

La pestaña de diseño permite observar y modificar las características de visualización del sistema de una forma muy rápida e intuitiva. De hecho, al arrastrar uno de los componentes desde la librería, este queda disponible pudiéndose modificar cualquiera de sus características (Catálogo de componentes completo en **ANEXO 3: LIBRERÍA APP DESIGNER**). En la siguiente figura se presenta una acción a modo de ejemplo de construcción de la parte visual.

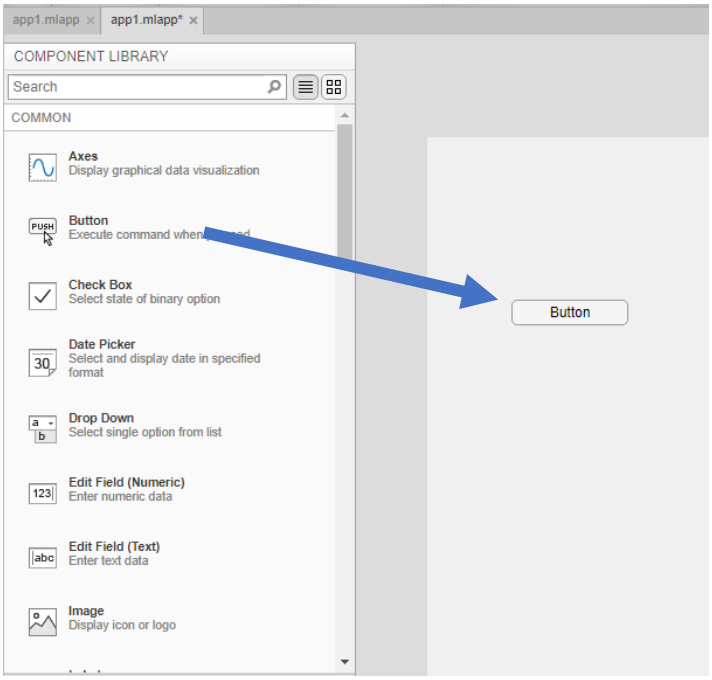


Figura 45.- Ejemplo de arrastrar componente

En la otra pestaña van apareciendo reflejados los cambios que se realizan en la parte de visualización, para así seleccionar el comportamiento de la aplicación en funcionamiento. De esta forma se puede transformar cualquier modificación (pulsar botones, editar campo, …). Esta parte es la encargada de que el programa cuente con las especificaciones principales.

Para este proyecto en concreto se ha decidido optar por la realización de una interfaz gráfica de usuario mediante el uso de esta última herramienta que recibe el nombre de *App Designer*.

I.7.7 COMPILACIÓN Y DISTRIBUCIÓN DE LA APLICACIÓN

Una vez diseñada e implementada la aplicación se procede a su encapsulamiento para su posterior distribución. En los siguientes apartados se presenta la información necesaria para comprender este proceso, así como los métodos o alternativas que existen para llevarlo a cabo.

I.7.7.1.- ¿QUÉ ES Y CÓMO FUNCIONA UN COMPILADOR?

Un compilador es un programa especial cuyo objetivo es traducir de código fuente a lenguaje de máquina. Esto quiere decir que transforman las líneas de texto o código en ficheros binarios, para que directamente sean interpretables por circuitos microprogramables, como sería el microprocesador de un ordenador o autómata. Con este proceso se obtiene un ejecutable (.exe) a partir de un programa en cualquiera de los lenguajes de programación. Este proceso se representa en la siguiente figura.

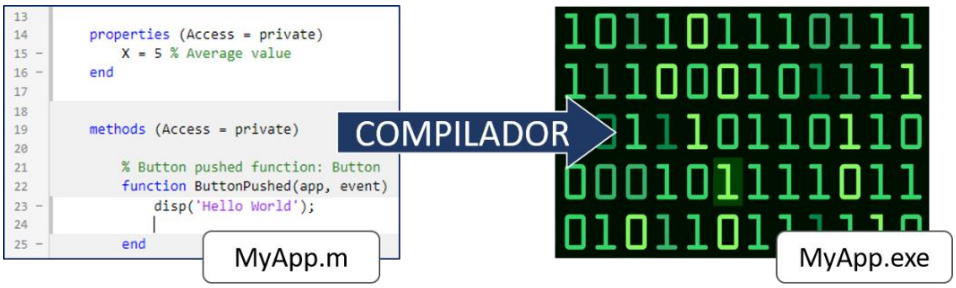


Figura 46.- Representación del funcionamiento de un compilador

El funcionamiento normal de un compilador se puede dividir en las siguientes fases:

1. Se lee el código fuente desde la memoria de la computadora.
2. Este código se transforma en código objeto o módulo de objeto. De hecho, un programa puede, y será lo habitual, estar formado por muchos objetos y librerías que deben ser unidas para poder empaquetarlos en un ejecutable.
3. Se realizan las conexiones y enlaces entre componentes del programa.
4. Los bloques de memoria son reasignados dentro del programa de modo que no se produzca solapamientos en la memoria.
5. Estos archivos compilados se graban en algún tipo de memoria permanente.
6. Como resultado final se obtiene un archivo o programa ejecutable.

Una vez presentada tanto la definición como el funcionamiento de un compilador, se da paso a analizar las alternativas posibles.

I.7.7.2.- MÉTODOS DE COMPILACIÓN EN MATLAB

Puesto que se ha realizado el diseño de la aplicación en Matlab, es necesario o muy recomendable utilizar alguno de los métodos de compilación propios de este software. Si bien es cierto, que históricamente ha sido un tema complicado, desde las últimas versiones ha ido mejorando este punto.

FUNCIONALIDADES MÁS RECIENTES EN MATLAB

La realización y empaquetamiento de aplicaciones es una de las asignaturas pendientes de Matlab, ya que muchas otras son mucho más utilizadas. Sin embargo, esta carencia ha hecho que se empiece a mejorar la funcionalidad del programa en este aspecto. A continuación, se presentan una serie de funcionalidades incorporadas recientemente:

- **Web apps**: Permite la configuración y despliegue de aplicaciones web mediante un flujo de trabajo mucho más simple que en versiones anteriores.
- **Compilador de aplicación**: A la hora de empaquetar en forma de aplicación independiente, se identifica automáticamente el tipo de datos de Matlab de las entradas de línea de comandos numéricas.
- **MATLAB Runtime**: este componente es un soporte para actualizaciones de Matlab.
    - **Azure HDInsight**: *MATLAB Runtime* está disponible para su distribución como una acción de script de Azure HDInsight.
    - **Apache Ambari:** *MATLAB Runtime* está disponible para su distribución como una pila de Apache Ambari.
    - **Cloudera:** también se puede descargar como un paquete con Cloudera Manager.

Seguidamente se describen y analizan las alternativas tenidas en cuenta a la hora de seleccionar la forma o formas finales de la aplicación del proyecto.

APLICACIONES INDEPENDIENTES O STANDALONE

Se entiende como **Aplicación independiente** o ***standalone software*** aquella aplicación completa, que se distingue de una extensión (o ***Add-on***) o de un plugin. Por lo tanto, hablar de una aplicación independiente quiere decir que se trata de un producto que se puede utilizar solo, sin necesidad de módulos o instalaciones complementarias. En este caso concreto, este tipo de aplicaciones no necesitan la instalación previa de Matlab para ser ejecutadas.

En la siguiente figura se muestra el esquema del empaquetamiento de este tipo de aplicaciones. *MATLAB Compiler* es la parte más importante del proceso, ya que es el encargado de traducir el código fuente al leguaje deseado.

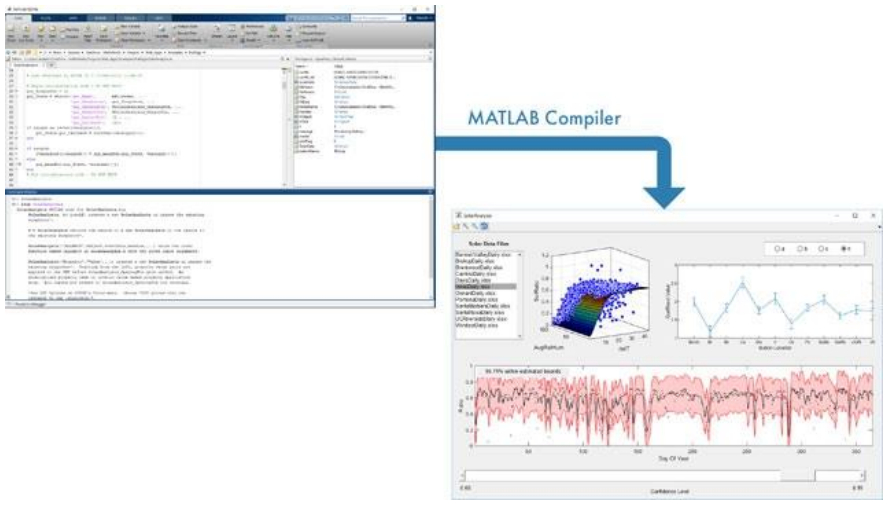


Figura 47.- Esquema de funcionamiento del compilador

Durante el proceso de compilación se encapsulan tanto la aplicación como el conjunto de librerías compartidas *MATLAB Runtime* que son las encargadas de la ejecución de aplicaciones y componentes de Matlab compilados. Desde las últimas versiones, el compilador cuenta con una interfaz que facilita el proceso y que además permite personalizar tanto la apariencia como las características del programa a compilar.

Como salida de este proceso de compilación se obtiene un fichero .exe que, al ejecutar, instala la aplicación diseñada. De esta forma, se convierte en un software fácilmente distribuible y accesible a la mayoría de los ordenadores.

WEB APPS

Esta opción permite empaquetar los programas de Matlab como aplicaciones web y compartirlas mediante una URL exclusiva. Esto permite la distribución de la aplicación sin necesidad de que el usuario deba instalar absolutamente nada.

El alojamiento de estas aplicaciones web se realiza a través de *MATLAB Web App Server*, un servicio que se pone a disposición con el pack de *MATLAB Compiler* [9]. Para representarlo se muestra la siguiente figura.

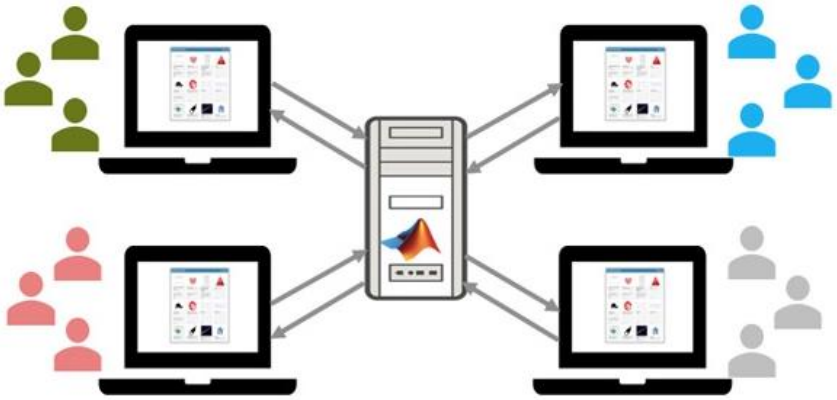


Figura 48.- Esquema de empaquetamiento app web con Web App Server

Una parte fundamental de este tipo de empaquetamiento es la configuración del *Server*, que será el lugar en el que se localicen las aplicaciones a desplegar mediante URL exclusiva. El montaje de este servidor requiere de una serie de pasos:

1. Buscar la carpeta que contiene el instalador de *MATLAB Web App Server*.
2. Ejecutar en modo administrador el instalador.
3. Verificar / Añadir MATLAB Runtime a la lista de variables del entorno de Windows (PATH).
4. Configuración avanzada del *Server*.

En cuanto a este último punto, en la figura siguiente se muestran las dos pestañas de configuración. Desde estas ventanas, se puede activar y pausar el sistema y por lo tanto permitir o no el acceso a las aplicaciones Web. Una vez activo el servidor, únicamente hay que comprobar que el archivo .ctf generado en la compilación de la aplicación se encuentra en la carpeta **apps** del *Web App Server*.

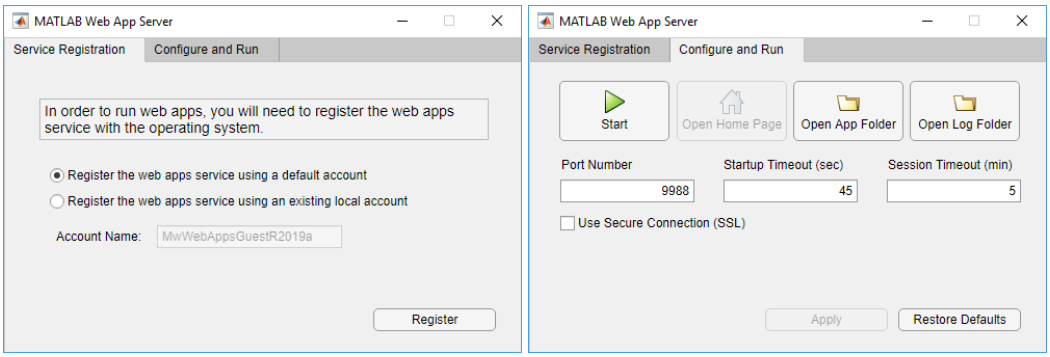


Figura 49.- Pestañas de MATLAB Web App Server

Una vez configurado el server, para acceder a las aplicaciones se pulsa el botón ***Open Home Page***. Seguidamente se abre en el navegador una pestaña con los accesos a los diferentes programas compilados y subidos al servidor. Esta página de inicio tiene el siguiente aspecto.

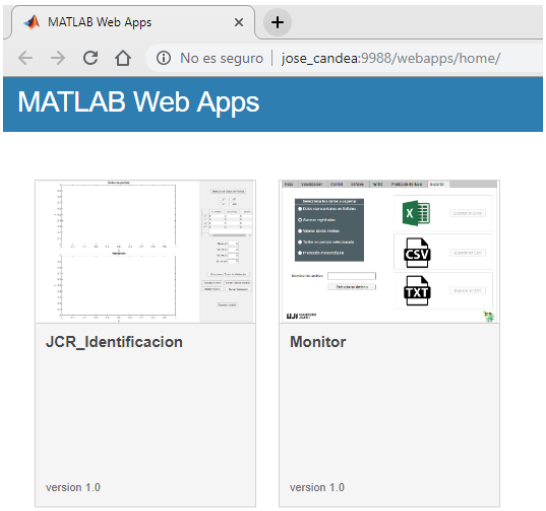


Figura 50.- Home Page del server de WebApps

Al hacer clic sobre alguna de las aplicaciones esta se ejecuta en el propio navegador, manteniendo todas sus funcionalidades.

COMPLEMENTOS DE MICROSOFT EXCEL

Esta opción permite compartir programas de Matlab como complementos de Microsoft Excel e integrarlos en Excel de forma que con solamente arrastrarlos y colocarlos en hojas de cálculo se ejecute el programa. Los usuarios no tienen necesidad de conocer Matlab para poder utilizar estos complementos.

A pesar de que esta es una gran alternativa para algunos casos, en el presente proyecto no se considera como óptimo ni adaptado a la situación.

HERRAMIENTAS MATLAB DE EMPAQUETADO

A la hora de empaquetar la aplicación, además de existir varios métodos de compilación también existen varias herramientas tanto dentro como fuera de Matlab. Sin embargo, puesto que el proyecto se ha realizado en Matlab, se decide utilizar la herramienta propia de este programa. En el siguiente esquema se puede observar la sencillez y facilidad con la que se puede

empaquetar una aplicación. En esta herramienta se puede seleccionar la forma final que se desea para la aplicación.

Figura 51.- Esquema de proceso de empaquetado de aplicación

Además, esta herramienta permite personalizar el proceso de instalación y la información que se muestra en los cuadros de dialogo, el icono, etc. En el recorte de pantalla que se muestra en la figura siguiente, se pueden observar los siguientes campos a rellenar:

- **Nombre** y **Versión** de la aplicación.
- **Mail** o **datos del desarrollador.**
- **Descripción** de la aplicación.
- **Icono de escritorio** e **imagen a mostrar durante la instalación.**
- **Documentos requeridos** para ejecutar la aplicación**.**

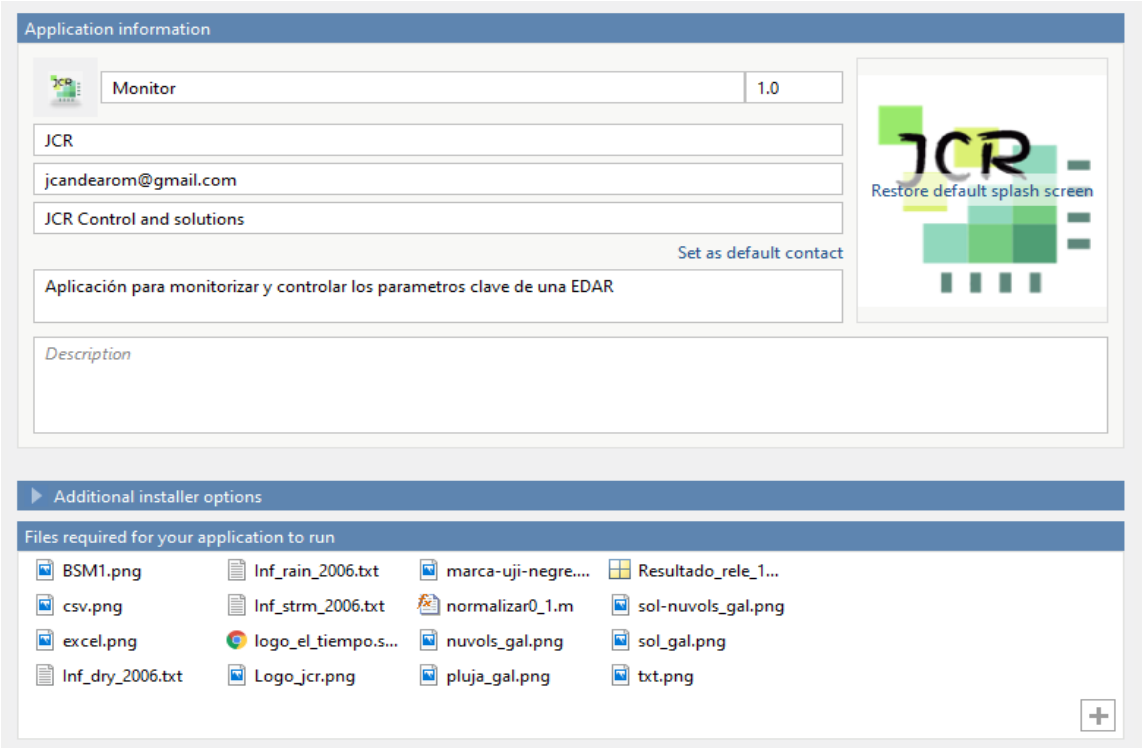


Figura 52.- Captura de la interfaz de compilación

I.8 RESULTADOS FINALES

En este apartado se procede a analizar los resultados obtenidos en cada una de las fases del proyecto. Por lo tanto, se divide en diferentes puntos para que sea más fácil su comprensión y lectura.

I.8.1.- RESULTADOS SOBRE EL SIMULADOR

El simulador final presenta la siguiente estructura: se parte de los ficheros generados al modelar el sistema con **WEST**, se pasa a **Matlab** y se editan de los ficheros a voluntad, se ejecutan los comandos para la simulación a través de **Tornado** y, finalmente, se extraen los resultados. En la siguiente figura, se muestra el esquema final de este simulador.

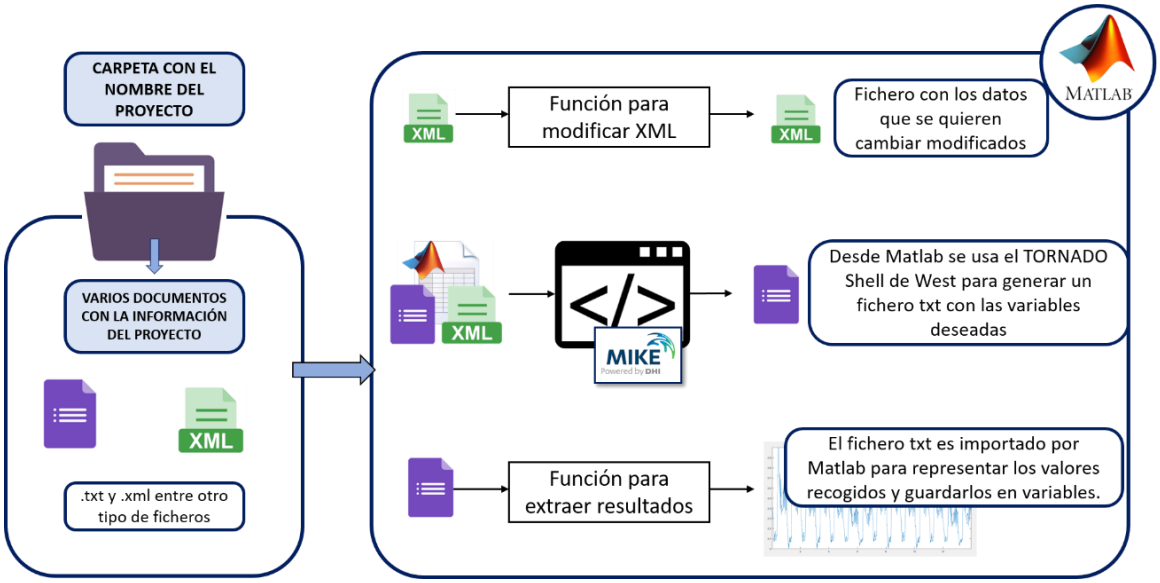


Figura 53.- Esquema final del simulador

Una vez se ha modelado el sistema y se ha realizado la conexión entre Matlab y Tornado, se lanzan una serie de experimentos para verificar el correcto funcionamiento y la validez de los resultados. Por un lado, se comprueba que los resultados son idénticos si se lanzan a través del simulador que si se hace por medio del propio software de simulación. Por otro lado, se corrobora que los resultados son los mismos que los recogidos en el BSM1. Finalmente, se debe validar que el resultado es exactamente el mismo si se simulan *X* días en una única simulación o concatenando simulaciones con una duración menor, esto es importante para la implantación del algoritmo de control.

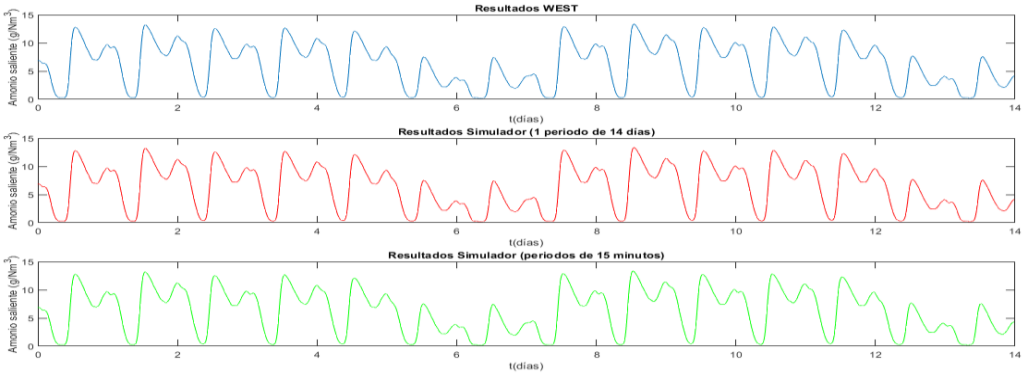


Figura 54.- Comparación de resultados de simulador

En la figura anterior, se puede ver claramente que los experimentos son equivalentes. Así pues, se da como valido el simulador considerando que los resultados obtenidos son correctos y, por lo tanto, equivalentes a los reales.

En termino de eficiencia de cálculo, se van a comparar tiempos de simulación realizando el mismo experimento, tanto en WEST como a través del simulador construido para este proyecto. En ambos experimentos se utilizan los mismos valores para las variables de entrada, $\mathbf{K_La} = 84d^{-1}$ y $\mathbf{Q_{int}}=55385\ m^3/d$.

Por un lado, el experimento lanzado desde el software de modelado tarda 9 minutos y 43 segundos en realizar una simulación de 14 días. En la siguiente figura, se muestra el centro de control de las simulaciones dinámicas en WEST.

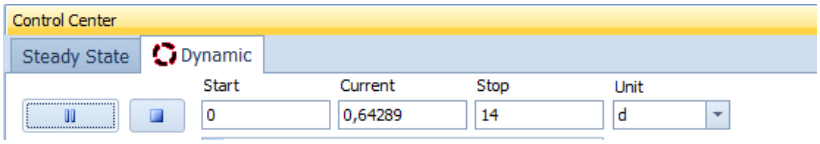


Figura 55.- Cuadro de lanzamiento de experimento en WEST

Por otro lado, si lanzamos el mismo experimento desde el simulador construido en Matlab. En este caso el programa tarda 542 segundos, es decir, 9 minutos y 5 segundos, mejorando la marca del test anterior. Para cronometrar este experimento se ha utilizado la función ***tic, toc*** como se muestra en la siguiente figura.

```
tic
[y1 , y2, u1, u2, wQ]=SimuladorWWTP_14dias(84,55385,1,'TEST_VELOCIDAD');
tiempo_total=toc;
```

Figura 56.- Código para realizar el test de velocidad

Con estas comprobaciones se demuestra que el simulador resultante de la integración de funciones y la conexión con Tornado es **eficiente** y los resultados obtenidos son **correctos** y **fiables**. A partir de este punto se puede dar paso a la siguiente fase del proyecto.

I.8.2.- RESULTADOS EXCITACIÓN DEL SISTEMA

La excitación del modelo sirve para conocer el **comportamiento del sistema** ante cambios en las variables de entrada, en este caso en la cantidad de aire que se introduce en tanque nº 5 (K_La) y el caudal de recirculación interna (Q_{int}). En los siguientes apartados se presentan los resultados de los diferentes métodos de excitación empleados.

RESULTADOS DE EXCITACIÓN ALEATORIA DEL SISTEMA

Para empezar, se excita el sistema con escalones aleatorios de las variables de entrada (u_1, u_2), para comprobar cuáles son sus efectos sobre el sistema. De esa forma se identifican las relaciones entre las variables que se han elegido de entrada y salida. A continuación, se presentan los resultados de la excitación aleatoria del sistema, fijando la otra variable para ver el efecto aislado de cada una de ellas.

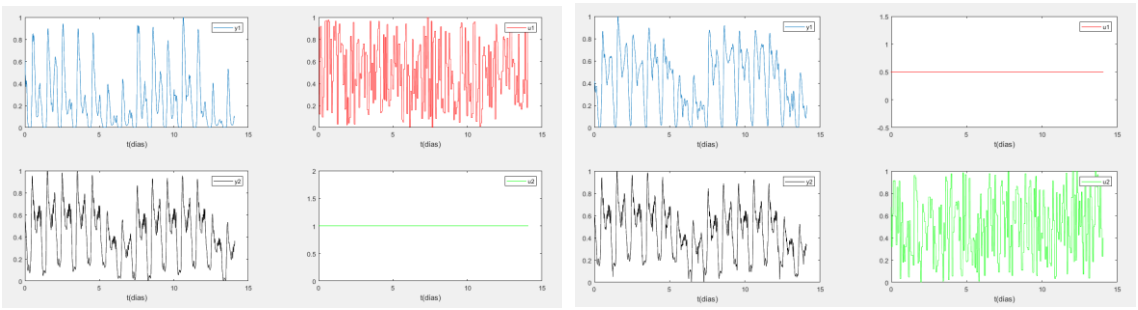


Figura 57.- Comparación de señales, con u_2 fija (izq.) - con u_1 fija(dcha.)

Las primeras conclusiones que se pueden extraer son:

1.- **Relación de las variables** – Gracias a esta primera excitación se observa que la variable u1 (K_La), tiene un gran impacto sobre y_1 (amonio), mientras que la variable u2 (Q recirculación) lo tiene sobre y_2 (salida de fangos).

2.- **Signo de la relación** – Además de las relaciones anteriores, los resultados obtenidos muestran que ambas relaciones son inversas. Es decir, el amonio aumenta cuando se disminuye el valor de K_La y, de igual modo, la salida de fangos disminuye al aumentar el caudal de recirculación interna. Los resultados obtenidos son coherentes ya que el K_La es un factor determinante en la fase de reducción del amonio; y en cuanto a la otra relación, al aumentar la recirculación se reduce el caudal de entrada del decantador y por tanto disminuye la salida de fangos.

En las siguientes imágenes se muestran ejemplos de las relaciones que se comentan en los puntos anteriores.

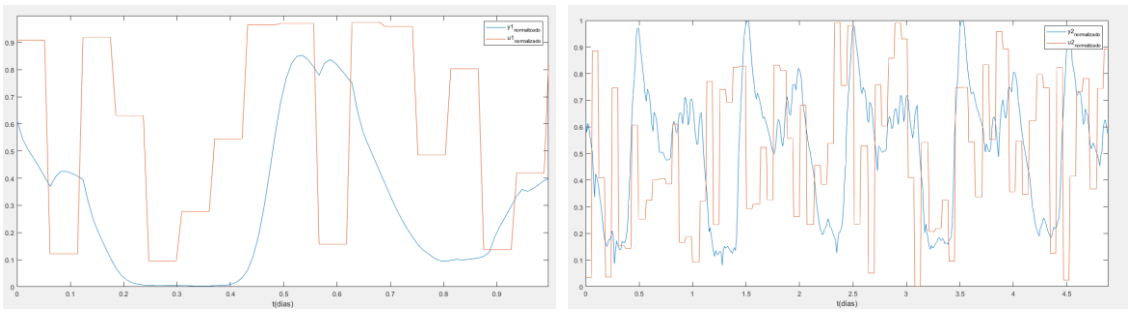


Figura 58.- Muestra de las relaciones entre las entradas y salidas

RESULTADOS DE EXCITACIÓN CONTROLADA DEL SISTEMA

Tras detectar la relación entre las variables y cómo se comportan frente a cambios de tipo escalón, se decide implementar un relé para que los cambios de las señales de entrada ya tengan relación con la forma deseada. Para empezar, es importante definir que un control tipo relé o todo/nada, se activa o desactiva la señal de entrada en función de la variable de salida realimentada.

Para que la excitación sea mayor, el relé implementado cuanta con una pequeña modificación, cada vez que se activa la señal de entrada es un valor aleatorio alrededor de un máximo y un mínimo fijados. Esto hace que se puedan estudiar diferentes situaciones de funcionamiento. Para ilustrar la variación que se ha comentado se presenta la siguiente gráfica. En ella se observa que a cada activación del relé su valor es diferente y constante.

$$u_{new} = u_{max\ /\ min} \cdot rand \cdot 0.15 + 0.9 \qquad (17)$$

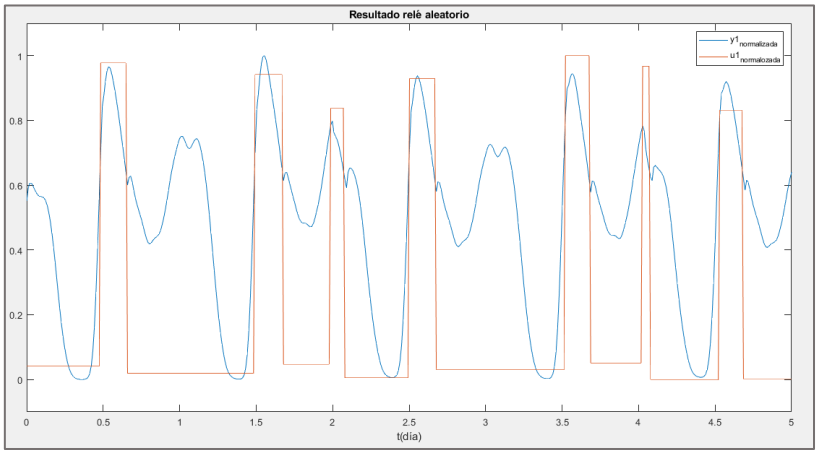


Figura 59.- Representación del RELÉ RAND

En cuanto a la decisión de incorporar histéresis en el control por relé, se han realizado una serie de experimentos. En estos se ha demostrado que en los casos en los que no se ha utilizado histéresis, el sistema realiza cambios muy bruscos para los actuadores que están trabajando. De hecho, tanto la bomba de recirculación como la aireación se basan en motores que se verían afectados por el constante cambio de consigna. La siguiente gráfica presenta la forma de las señales en caso de trabajar con un relé sin histéresis.

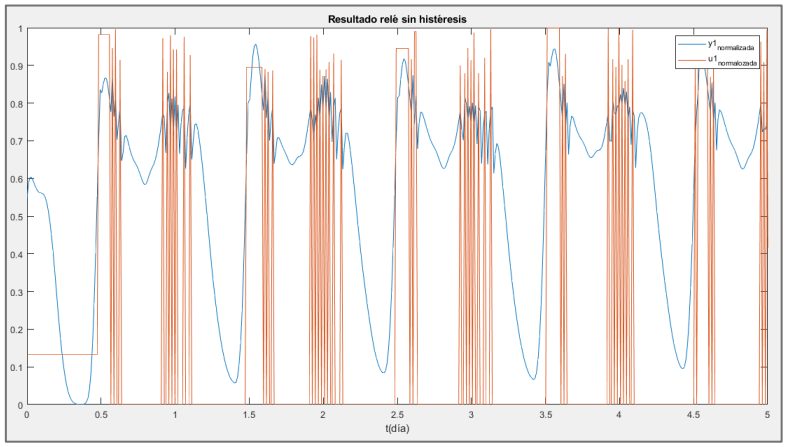


Figura 60.- Representación de relé sin histéresis

I.8.3.- RESULTADOS SOBRE LA IDENTIFICACIÓN DEL MODELO

Una vez excitado el modelo y observando las relaciones entre las variables, se da paso a la identificación del modelo. De todas las opciones presentadas en el apartado **I.7 ANÁLISIS DE SOLUCIONES**, se han ido probando diferentes configuraciones, ordenes, *delays*, no linealidades y grados de las variables hasta hallar el modelo más ajustado. Para establecer de una forma y clara el funcionamiento del modelo en la siguiente figura se presenta su esquema.

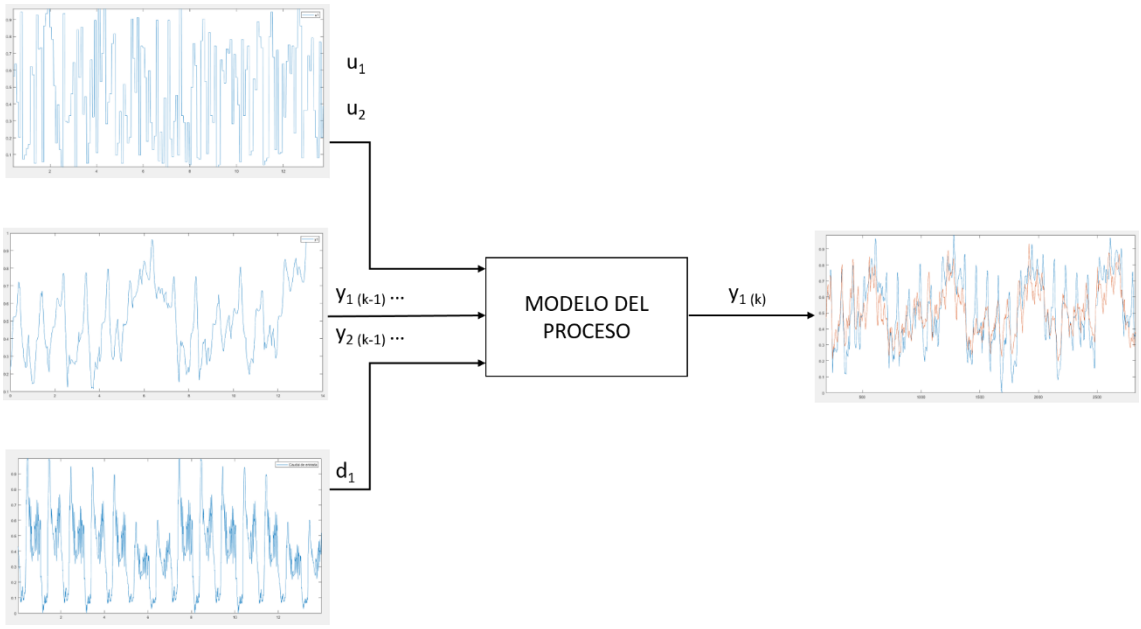


Figura 61.- Esquema final del modelo

Para decidir estos parámetros del modelo se han realizado una serie de pruebas y ensayos. En la siguiente figura se muestra la relación que existe entre el orden del regresor, el error y el tiempo de cálculo. En esta gráfica se puede ver como no por aumentar el orden, el modelo hace predicciones cada vez mejores. De hecho, llega un momento en el que el modelo deja de converger. En este proyecto se observa que con **un orden de alrededor de 6 o 7 es suficiente**.

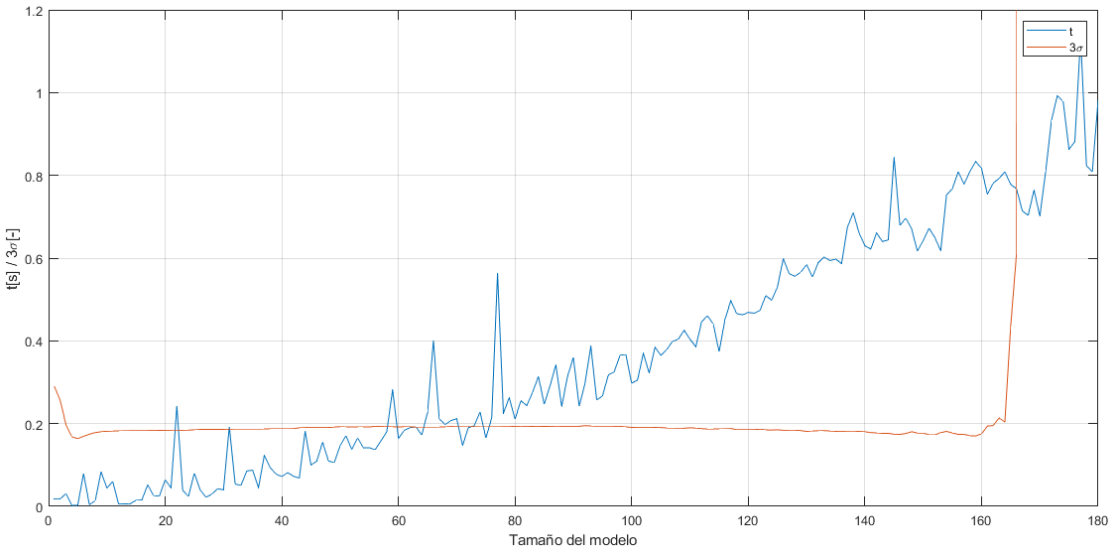


Figura 62.- Representación error y tamaño del modelo

Aunque el orden del modelo es relativamente bajo, al tratarse de un sistema claramente multi-variable se ha optado por dividir este, es decir, asignar un orden a cada una de estas variables. De esta forma, se puede dar mayor peso a las variables que se consideren más representativas del sistema.

De la misma forma, se han asignado *delays* diferentes a cada variable, con los cambios en la identificación y en la construcción del algoritmo de control que estas modificaciones conllevan. Por lo tanto, los parámetros a determinar se recogen en la siguiente tabla.

$u_1 (K_La)$	$u_2 (Q_{int})$	$y_1 (S_{NH})$	$y_2 (M_{fangos})$	$d - wQ (Q_i)$
n_{u1}	n_{u2}	n_{y1}	n_{y2}	n_{wQ}
d_{u1}	d_{u2}	d_{y1}	d_{y2}	d_{wQ}

Tabla 8.- Parámetros del modelo multivariable

En la siguiente figura, se presentan los resultados de la identificación, en esta gráfica se muestra en rojo la predicción y en azul los datos reales con los que se ha diseñado el modelo. En ella se observa claramente como la predicción va siguiendo los valores de los datos reales. Sin embargo, el proceso de validación aún no ha concluido.

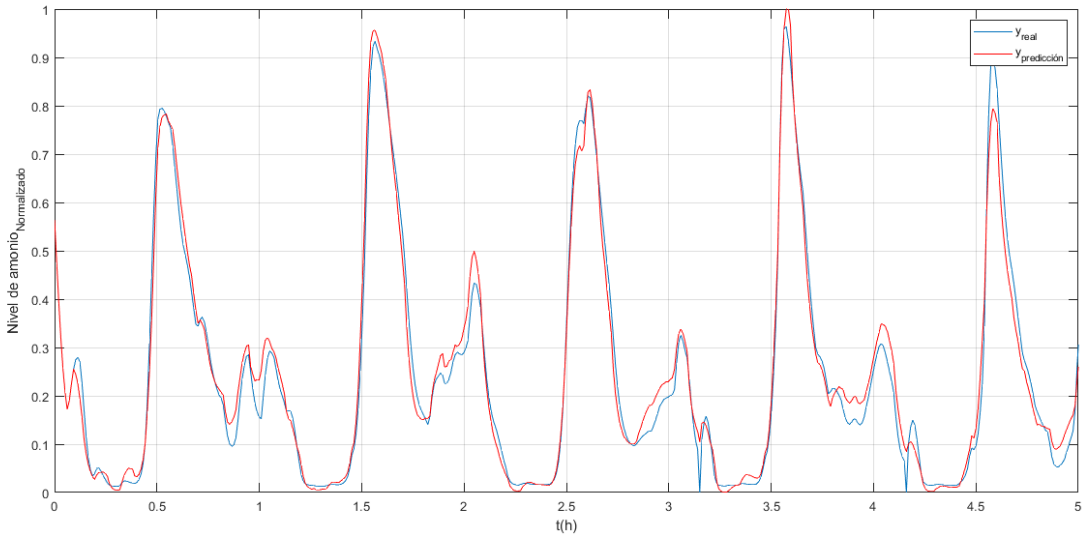


Figura 63.- Representación de los datos reales y las predicciones (datos iniciales)

A continuación, se muestra la gráfica de validación del modelo, es decir, se seleccionan unos datos diferentes a los usados en la creación del modelo y se hace la predicción utilizando el modelo construido precedentemente. De nuevo se observa que la predicción es considerablemente buena, con un rango de **3σ** de aproximadamente el 20%.

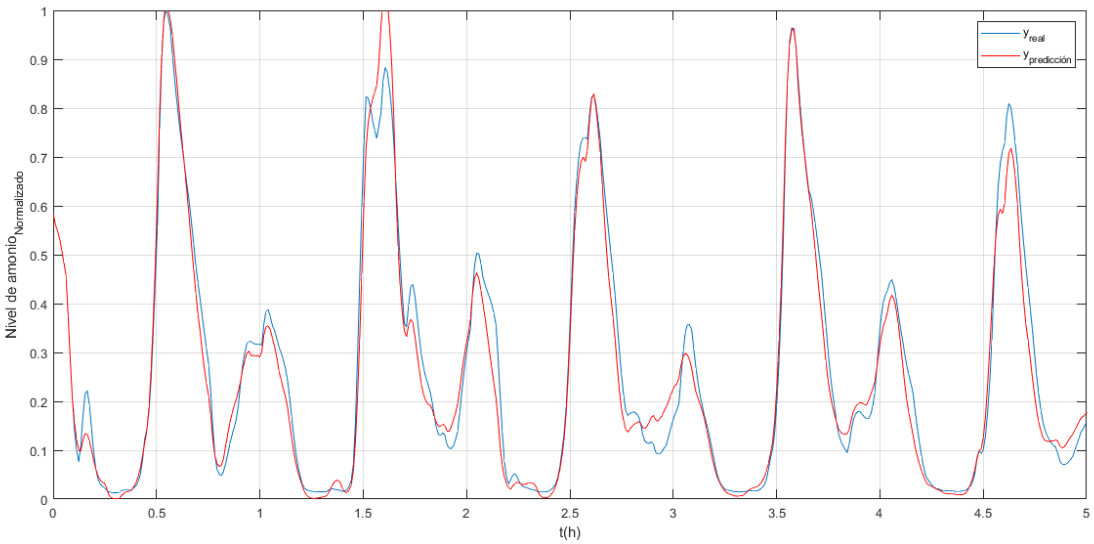


Figura 64.- Representación de los datos reales y las predicciones (validación)

Tras una serie de pruebas y analizando las diferentes opciones propuestas, para este proyecto el modelo queda de la siguiente forma.

Ordenes – $n=[n_{y_1}, n_{u_1}, n_{y_2}, n_{u_2}, n_{wQ}]$	[5 , 5 , 3 , 3 , 5]
Delays – $d=[d_{y_1}, d_{u_1}, d_{y_2}, d_{u_2}, d_{wQ}]$	[1 ,2 , 2, 1 , 3]
N.L.	$[\, y_1^{0.5} \,, 1 \,, 1 \,, 1 \,, 1 \,]$
Grados – $g=[g_{y_1}, g_{u_1}, g_{y_2}, g_{u_2}, g_{wQ}]$	[1 , 1 , 1 , 1 , 3]
Modelo final	
$y_k = \sum_{i=1}^{n_{y_1}} (a_i \cdot y_{1_{k-i}}) + \sum_{i=1}^{n_{y_2}} (b_i \cdot y_{2_{k-i}}) + \sum_{i=1}^{n_{u_1}} (c_i \cdot u_{1_{k-i}}) + \sum_{i=1}^{n_{u_2}} (d_i \cdot u_{2_{k-i}}) + \sum_{i=1}^{n_{wQ}} (e_i \cdot wQ_{k-i} + f_i \cdot wQ_{k-i}^2 + g_i \cdot wQ_{k-i}^3)$	

Tabla 9.- Datos del modelo predictor MPC

I.8.4.- IMPLEMENTACIÓN FINAL DEL ALGORITMO DE CONTROL

Con todo lo expuesto en el análisis de soluciones la estructura final de la **simulación del control MPC**, consta de las fases que se recogen en el siguiente esquema:

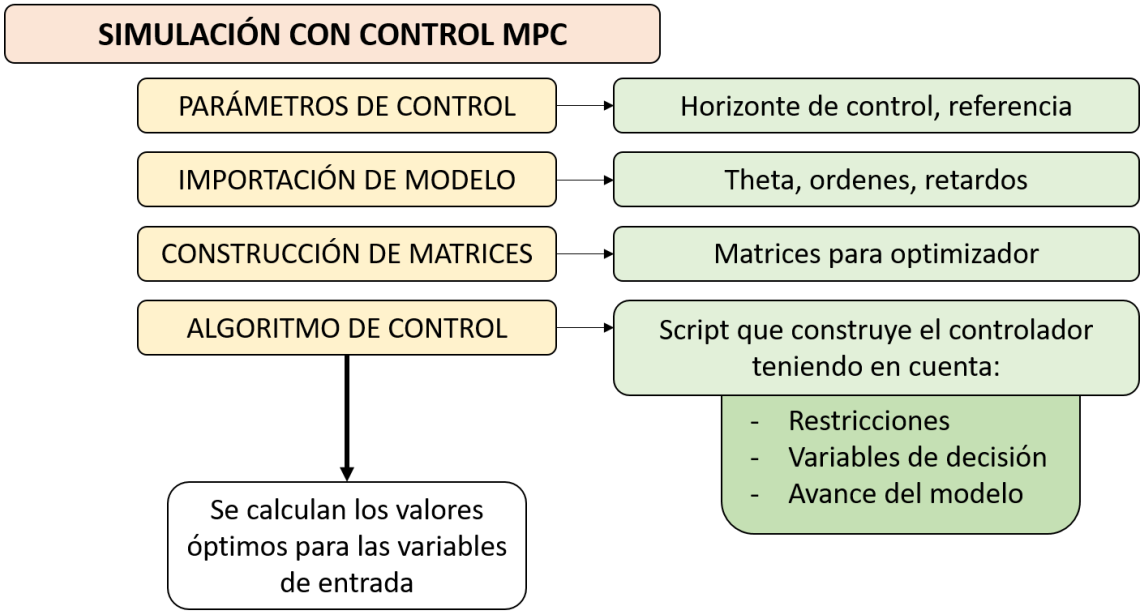


Figura 65.- Esquema de las etapas para la simulación con MPC

En el primer paso, se declaran los parámetros que sirven de base al algoritmo de control, el *Horizonte de predicción* y *la referencia o límite de la salida*. Tras varias pruebas, se han obtenido las siguientes conclusiones:

- El **Horizonte debe ser suficientemente grande** para que el modelo pueda asumir los cambios con un tiempo suficiente, de modo que sus reacciones se puedan observar a la salida.
- Cuanto mayor sea el Horizonte de control, más tiempo tarda el optimizador en llegar a una solución.
- La referencia o límite debe estar dentro de un rango posible, es decir, si se elige un límite demasiado bajo, el optimizador no es capaz de obtener resultados.

El siguiente paso importa el modelo desde un fichero *.mat* en el que previamente se han guardado las variables características del modelo de predicción. De este modo, el programa principal tiene acceso a los datos de identificación y puede comunicárselos a la siguiente fase del proceso.

Así pues, una vez conocida la información sobre la identificación y los parámetros de control, se procede a la construcción de las matrices (explicadas en el apartado **I.7.4.3.- MATRICES PARA EL OPTIMIZADOR**) utilizadas en el optimizador.

Una vez definidos todos los parámetros y variables anteriores se da paso a la creación del algoritmo de control. Es decir, se fija la función objetivo y las restricciones del caso de control concreto que se esté estudiando. También, se programa de forma que cada vez que se lanza el algoritmo este va actualizando tanto el vector de estados como el resto de los parámetros.

I.8.5.- RESULTADOS SOBRE EL CONTROL DEL SISTEMA

En este punto se recogen los resultados de las simulaciones en las que se ha incorporado el control predictivo basado en modelo. En la siguiente tabla se muestran información sobre los parámetros más importantes de los experimentos.

Modelo utilizado	Se han realizado pruebas con diferentes modelos y el seleccionado es el que aparece en la **tabla 9**.
Horizonte de predicción	Se considera parámetro clave y se estudia cuál es su valor óptimo.
Periodo de muestreo	Se estudia su influencia siempre guardando la coherencia con la elección en el simulador.
Límite de amonio a la salida	En este proyecto se establece un límite de amonio de entre **9-10 g/m^3**, considerando violación de la premisa cuando el amonio medio diario supere este valor.
Gestión de los no éxitos del optimizador	Cuando el optimizador no alcanza unos valores capaces de minimizar la función y cumplir las restricciones, lanza un mensaje de error. Es vital controlar este tipo de situaciones
Solver utilizado	Puesto que el *solver* tiene una gran influencia sobre el optimizador, se realizan y analizan diferentes opciones para construir un ranking con los seleccionados.

Tabla 10.- Parámetros de control

Para determinar estos parámetros característicos del control, se realizan una serie de experimentos. A continuación, se enumeran los resultados obtenidos entorno a estas pruebas.

- **INFLUENCIA DEL MODELO DE PREDICCIÓN.** Como ya se ha comentado previamente, el modelo es una de las partes fundamentales en el proyecto e influye de una forma considerable sobre la precisión del controlador **MPC**. De hecho, se debe seleccionar correctamente tanto su tamaño como sus parámetros para poder predecir los efectos de nuestras acciones sin cargar en exceso al optimizador.

- **PERIODO DE MUESTREO.** El periodo de muestreo es el tiempo que transcurre entre muestras de la simulación, es decir, es equivalente al número total de simulaciones que se van a realizar para llevar a cabo un experimento completo. La principal influencia de este parámetro es sobre la carga computacional del propio simulador, ya que a menor periodo de muestreo mayor número de llamadas y por lo tanto más lenta es la ejecución del experimento.
  Para este proyecto, se ha seleccionado un periodo de cálculo de **15 min – 0.0104 días**, esta decisión viene motivada por el tipo de datos de entrada del experimento del BSM1.

- **LÍMITE DE AMONIO.** Este parámetro tiene una gran influencia en el modelo, ya que es el que determina su comportamiento y condiciona las activaciones de las acciones de control. Como ya se ha comentado en este proyecto se selecciona un límite de **9 g/m^3**.

- **NO ÉXITOS DEL OPTIMIZADOR.** Puesto que los problemas de optimización no siempre tienen solución, es importante tener en cuenta estas situaciones. Para asegurar el correcto funcionamiento de esta estructura se colocan varios controladores en paralelo. De este modo, se consigue aumentar la seguridad del algoritmo, ya que, si uno de ellos falla, se activa uno de los otros. En la siguiente figura se muestra el esquema del sistema resultante.

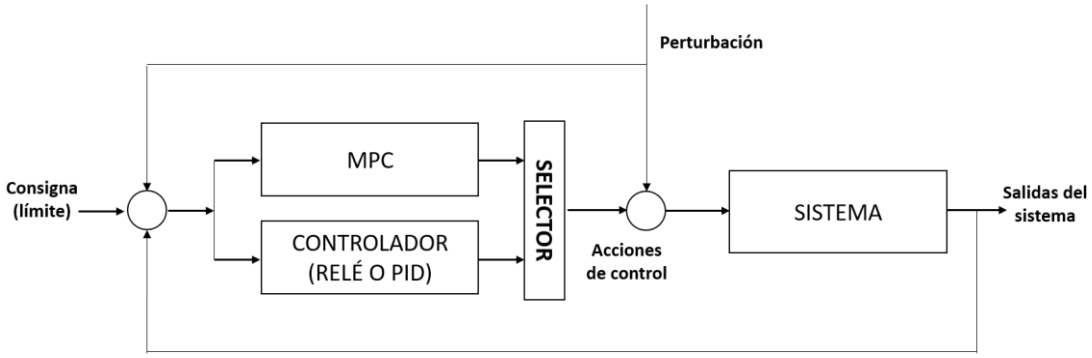


Figura 66.- Estructura de controladores en paralelo

Con esta estructura se consigue que el sistema este siempre controlado y que funcione de forma óptima.

- **INFLUENCIA DEL HORIZONTE DE PREDICCIÓN.** El Horizonte de predicción es uno de los principales motivos de la ralentización y del "no éxito" del proceso de optimización. Sin embargo, este parámetro es el encargado de definir los datos que se van a predecir.
  Por lo tanto, alrededor de esta variable entran en conflicto la carga computacional y el grado de previsión del controlador. Para visualizar el efecto de este parámetro, en la siguiente figura se representa la Evolución del tiempo de cálculo del optimizador en función del Horizonte de Predicción seleccionado.

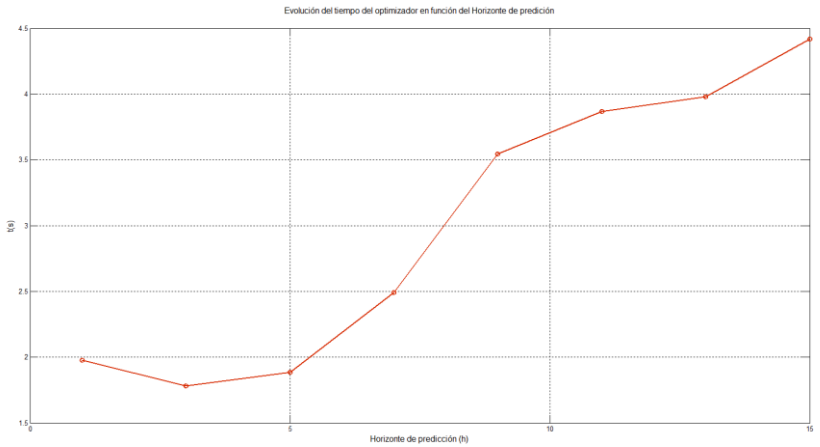


Figura 67.- Gráfica de Evolución entre tiempo y HP (Horizonte de predicción)

- **INFLUENCIA DEL *SOLVER* UTILIZADO.** Cada uno de los *Solvers* seleccionados tiene unas características de rendimiento distintas, por lo tanto, la mayor diferencia entre la utilización de uno u otro tendrá consecuencias sobre la duración del experimento. De hecho, al tratarse de problemas con carga computacional elevada, en términos de velocidad y tiempo de cálculo, el *solver* es uno de los puntos más importantes. En la

siguiente tabla se muestra un ranking en función de los tiempos de cálculo, como sinónimo del rendimiento e índice de desempeño.

RANKING DE RENDIMIENTO
1 PENBMI
2 MOSEK
3 SeDuMI
4 SDPT3

Tabla 11.- Ranking de rendimiento de los solvers

Todos los optimizadores han pasado el filtro de los test establecidos. De hecho, en la siguiente figura se muestran los resultados obtenidos para **SeDuMi** y **SDPT3**, se observa que la diferencia de tiempo de cálculo entre ambos es notoria pero aceptable y que, a pesar de ser los peores situados en el ranking, son lo suficientemente bajos para ser considerados como aptos.

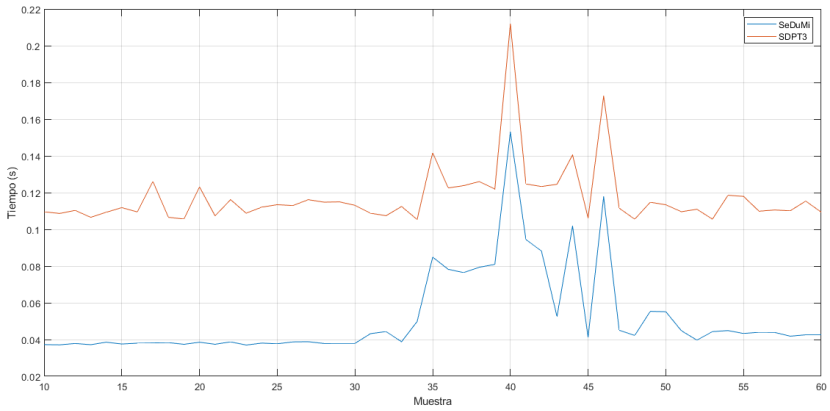


Figura 68.- Comparativa de tiempos entre SeDuMi y SDPT3

Por un lado, se realiza la comprobación de que el algoritmo predice correctamente, es decir que se adelanta a los acontecimientos. Para verificar esto, simplemente hace falta observar la gráfica siguiente en la que a pesar de que el valor de amonio a la salida aún está lejos de sobrepasar el límite marcado, la acción de control empieza a modificar el valor de la entrada para tener el tiempo suficiente para regular la salida. Esto es un claro ejemplo del funcionamiento de un control predictivo.

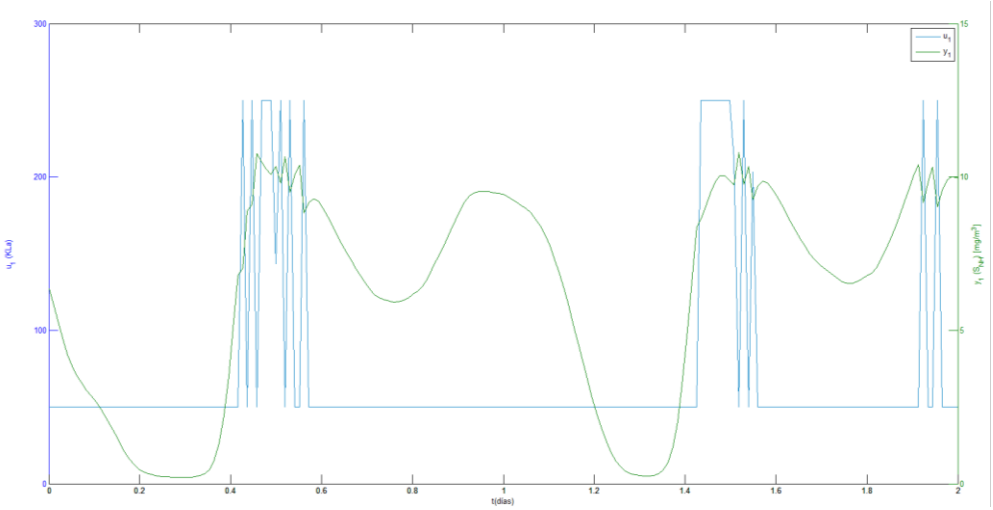


Figura 69.- Primeros resultados del control predictivo

Por otro lado, para evaluar su rendimiento se comparan los resultados obtenidos aplicando el **control predictivo** con los resultados del **relé**. En las siguientes gráficas, se representan los resultados de cada uno de los experimentos.

Primero, se muestra los resultados del relé, en los que a simple vista se observa que los límites de amonio son superados en numerosas ocasiones.

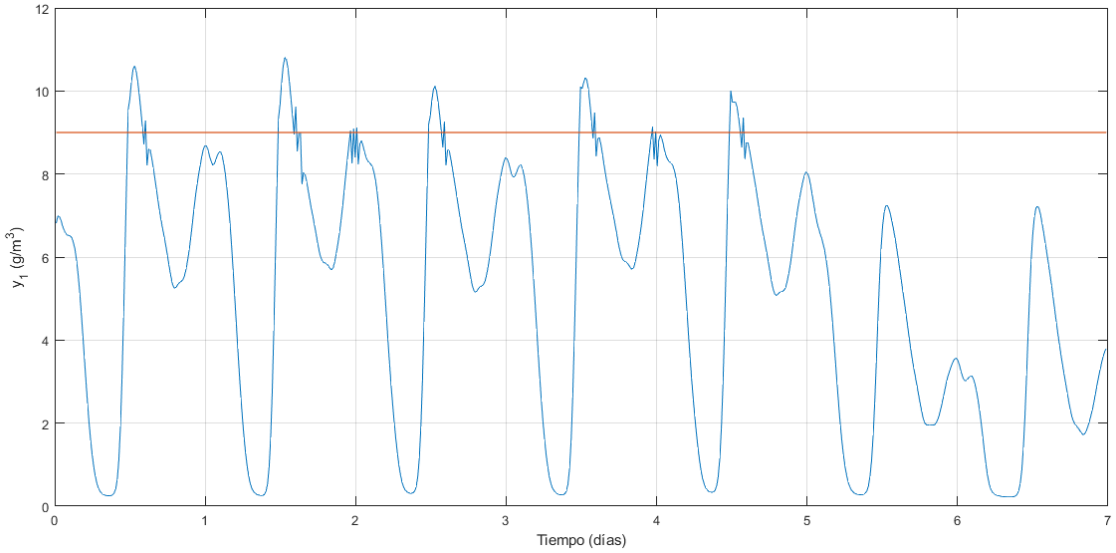


Tabla 12.- Resultados de aplicar el control RELÉ

Por su parte el control MPC, da como resultado la siguiente grafica en la que se observa que los valores pico de la señal han disminuido considerablemente.

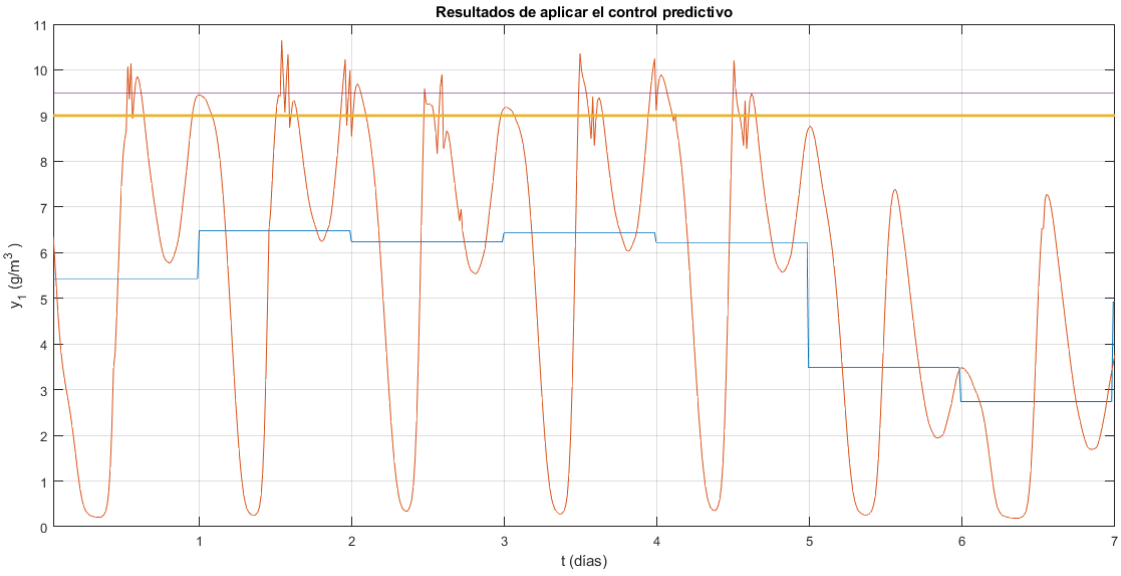


Figura 70.- Resultados de la de aplicar MPC

En términos energéticos, una parte fundamental es el tiempo que ha estado en funcionamiento el soplador. A continuación, se comparan ambos los periodos de ambos casos. Además, es importante destacar que la activación de esta acción de control en el MPC se produce en los periodos en los que la electricidad es más económica.

CONTROL	TIEMPO EN FUNCIONAMIENTO
Relé - On/Off	12 periodos – 3h (en 1día)
MPC	9 periodos – 2h y 15 min (en 1día)

Tabla 13.-Resumen resultado de controles

Para visualizar los resultados se han representado gráficamente las acciones de control calculadas con cada algoritmo. En esta figura, se puede observar como la señal del MPC se adelanta por dos motivos, para conseguir llegar al límite marcado y para evitar el periodo donde el precio de la electricidad es mayor.

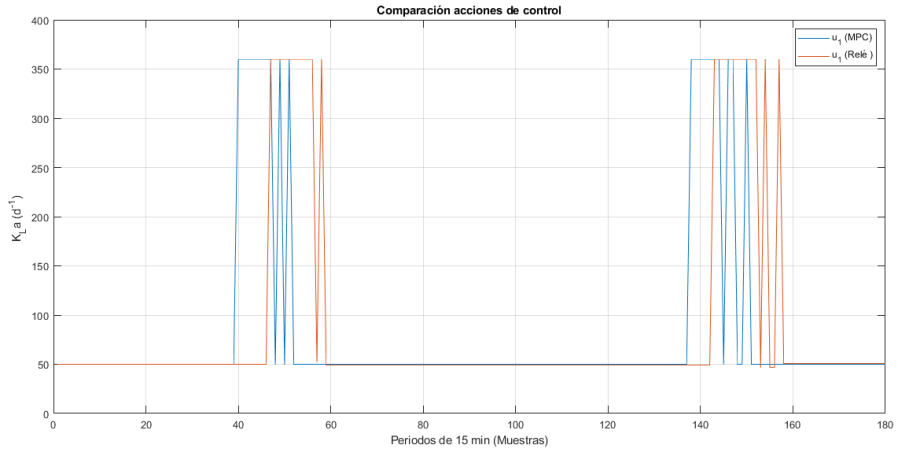


Figura 71.- Comparación acciones de control Relé y MPC

En la siguiente gráfica se representa el coste de la electricidad y los puntos de activación en función de la estrategia de control aplicada. En ella, se puede observar como el MPC activa el soplador antes de que el precio de la electricidad aumente, mientras que, el relé no tiene en cuenta esta condición. La tarifa seleccionada es la 6, para más información consultar **ANEXO 4,** modalidad de 6 períodos.

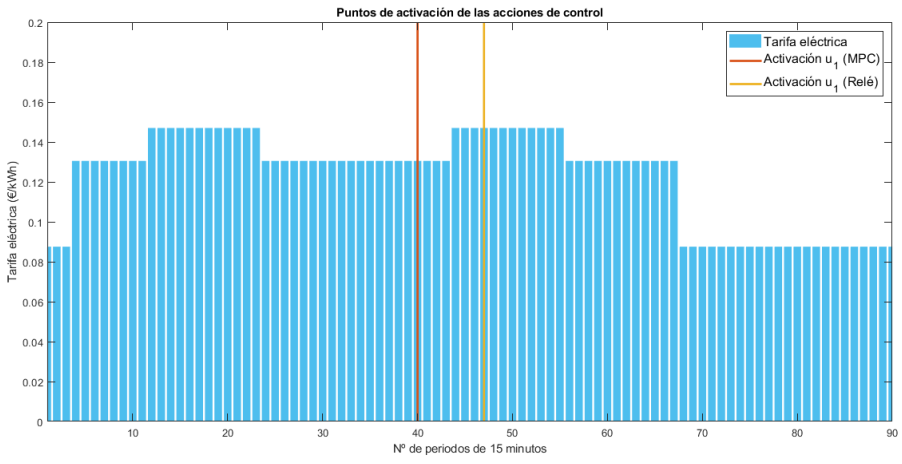


Figura 72.- Representación de los puntos de activación de las acciones de control

En cuanto al caudal de recirculación (u_2) se desprecia su influencia en términos de consumo, considerando que su influencia es mucho menor que la del soplador (u_1). Los cálculos sobre ahorro económico y energético se amplían en el apartado **I.11.2.- VIABILIDAD ECONÓMICA.**

I.8.6.- ESTRUCTURA Y FUNCIONAMIENTO DE LA APLICACIÓN

En este apartado se presenta con un grado de detalle considerable la estructura de la aplicación. La aplicación esta agrupada de en pestañas, a través de las cuales el usuario va navegando libremente (en función del permiso con el que cuente), cada una de las pestañas permite realizar una serie de acciones que se describen a continuación. La instalación de la aplicación se explica en el apartado **III.6.- MANUAL DE INSTALACIÓN DE APLICACIÓN DE CONTROL.**

PESTAÑA 1 – CONTROL DE ACCESO

La primera ventana que aparece al ejecutar la aplicación es la de identificación del Usuario. En esta pantalla, se pide el nivel de acceso y su correspondiente contraseña. Mientras no se introduzcan los datos correctos, la aplicación no permite el acceso al resto de pestañas. Además, aparecen una serie de mensajes que indican si la aplicación está bloqueada o el modo de acceso al que se ha accedido.

En las figuras que se muestran a continuación se observa la lista desplegable en la que se puede elegir el nivel de autorización, el campo en el que se introduce la contraseña y un botón para cerrar la sesión abierta. Al lanzar la aplicación esta aparece como bloqueada, mostrando una etiqueta en la que se lee “Aplicación bloqueada” (figura de la izquierda), al introducir los datos de acceso correctamente se desbloquea la aplicación. En función del nivel de autorización asignados al usuario, se activa un modo concreto de funcionamiento, por ejemplo, al entrar como “Operario” se indica “Modo Operario” (figura de la derecha).

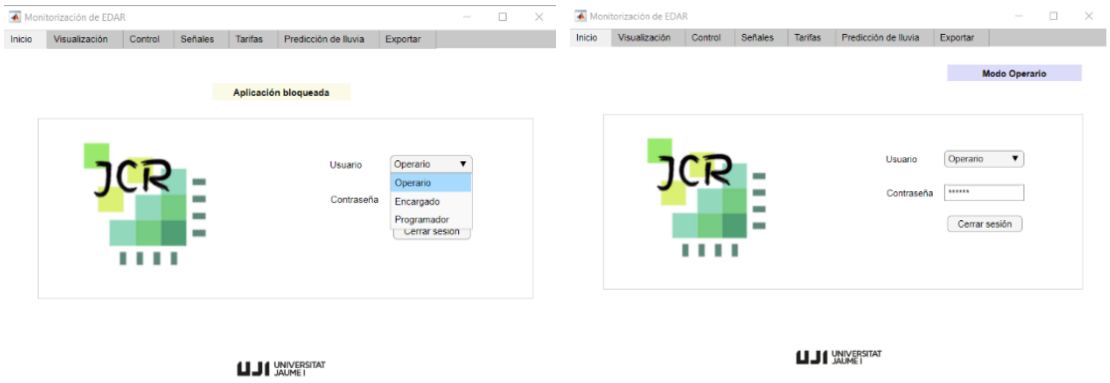


Figura 73.- Lista de tipos de usuarios y modo operario

Cada uno de los niveles de usuario, activa una serie de funcionalidades diferentes, ya que se considera que estos niveles son equivalentes a su jerarquía en la cadena de mando y por lo tanto la información que puede consultar o modificar varía. En la siguiente tabla se resume los permisos en función del nivel de Usuario.

NIVEL DE AUTORIZACIÓN	PERMISOS
Operario	- Acceso a pestaña de Visualización y Control - Sin permiso de escritura
Encargado	- Acceso a todas las pestañas - Permiso de escritura
Programador	-Acceso total al código

Tabla 14.- Resumen de niveles y permisos

Finalmente, en caso de introducir una contraseña incorrecta, la aplicación muestra un mensaje de error en el que se notifica que se ha producido un fallo a la hora de registrarse. En la siguiente figura se presenta el mensaje de error.

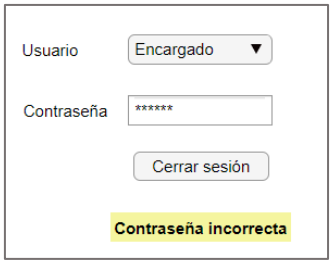


Figura 74.- Mensaje de Contraseña incorrecta

PESTAÑA 2 – VISUALIZACIÓN

En esta pestaña, se presenta una imagen del proceso que se está monitorizando, con una serie de etiquetas que permiten conocer en todo momento el estado de cada uno de los sensores (variables medibles en el caso del simulador), además de una serie de indicadores que cambian de color en función de si el valor es apto o no.

Por otro lado, esta pestaña cuenta con una tabla que va recogiendo las alarma que crea el sistema. Por ejemplo, si se supera el nivel de amonio a la salida de la EDAR, en esta tabla queda registrado el momento y el valor exacto.

También cuenta con botones de *Start/Stop* que permiten conectar y desconectar el sistema y con un reloj/calendario en el que se muestra la hora y la fecha actuales (en el caso del simulador, se irá actualizando a la fecha de simulación).

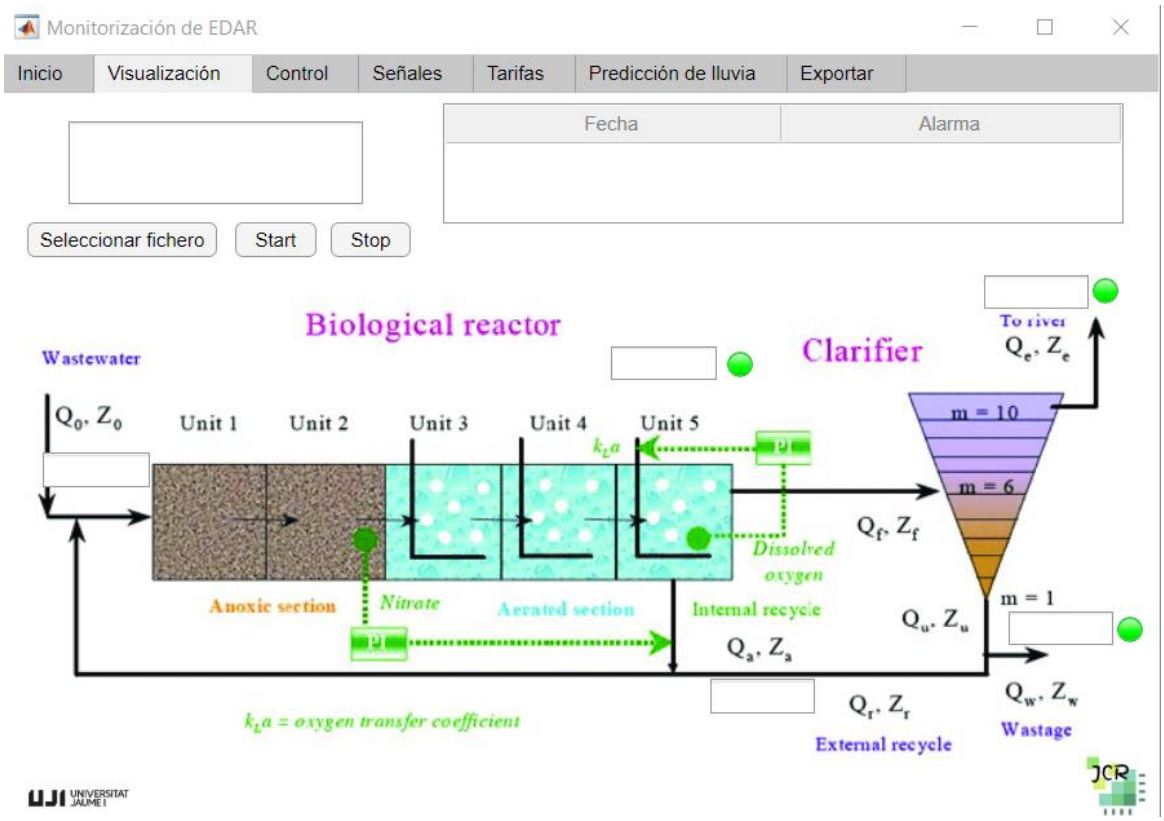


Figura 75.- Captura de pantalla de la ventana de visualización

Para interpretar el código de colores y alarmas, se recoge la información importante en la siguiente tabla:

Descripción del mensaje	Modificable	Notificación
[*FECHA Y HORA*] Se ha superado el límite de amonio [*VALOR*]	Sí	En pantalla y por mail
[*FECHA Y HORA*] Caudal de fangos muy elevado [*VALOR*]	Sí	En pantalla y por mail
[*FECHA Y HORA*] Soplador mucho tiempo al máximo	Sí	En pantalla

Tabla 15.- Listado de Alarmas

Además, se ha incorporado un botón que permite la selección del fichero a representar, pensando sobre todo en la etapa de simulación o análisis de resultados. Al pulsar sobre este botón, se abre una ventana de dialogo que accede al explorador de archivos y permite seleccionar el archivo independientemente de su ubicación.

PESTAÑA 2 – *CONTROL*

Esta pestaña es complementaria a la anterior, y muestra los datos en directo del nivel de Amonio en a la Salida. Sin embargo, en este caso se trata de una representación graficada permitiendo observar la tendencia de esta señal. Asimismo, incorpora una serie de indicadores de aguja que permiten conocer con exactitud el valor actual de cada variable.

En este caso, la pestaña incluye una tabla a modo de resumen diario, en la que se presentan la fecha, la concentración media de Amonio a la salida, la potencia media de aireación (K_La) y el consumo de ese día en concreto.

También se ha incluido un botón para vaciar la gráfica con el fin de regular la cantidad de datos mostrados y un selector de velocidad que permite acelerar el transcurso del tiempo de la simulación.

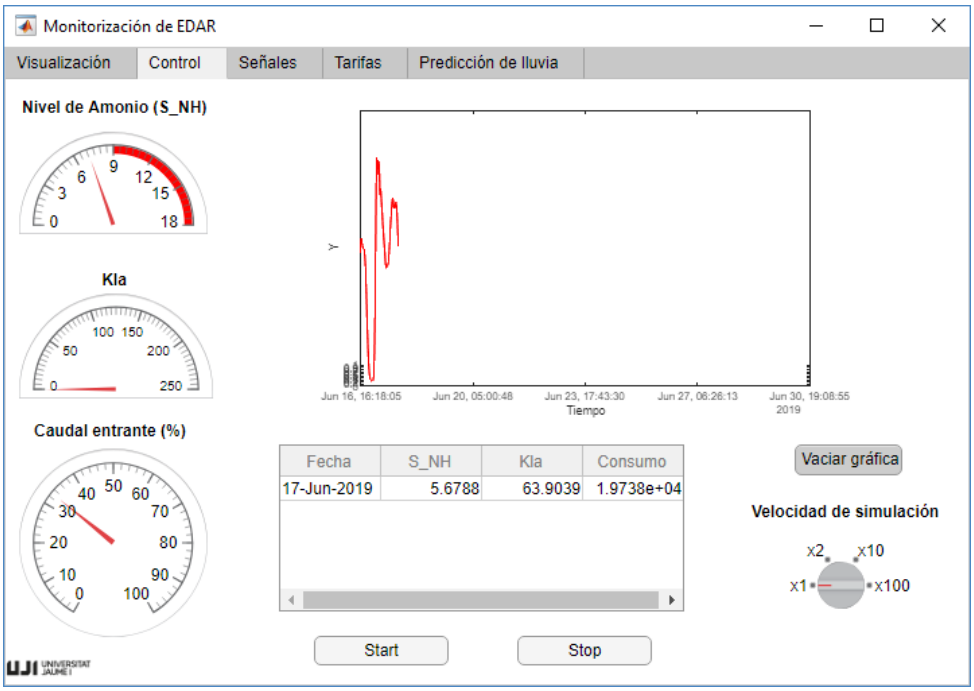


Figura 76.- Captura de pantalla de la pestaña de Control/Monitorización

PESTAÑA 3 – *SEÑALES*

Esta pestaña deja de lado el *streaming* o directo, para dar paso a los históricos y análisis de datos obtenidos hasta el momento. En la parte derecha de esta ventana, se permite seleccionar las señales que se desean visualizar.

El funcionamiento del selector es simple: al pulsar sobre una de las casillas junto a las etiquetas la señal a la que hace referencia esta etiqueta se muestra en la gráfica, de este modo se pueden seleccionar tantas señales como se desee, para que una de las señales desaparezca de la gráfica, simplemente habrá que deseleccionar la señal en cuestión.

Además, esta pestaña cuenta con un botón que permite "Normalizar" las señales que se están representando. Esta normalización se realiza con el fin de hacer las señales comparables. Por último, la aplicación cuenta con un botón "Vaciar gráfica" que permite borrar todas las señales.

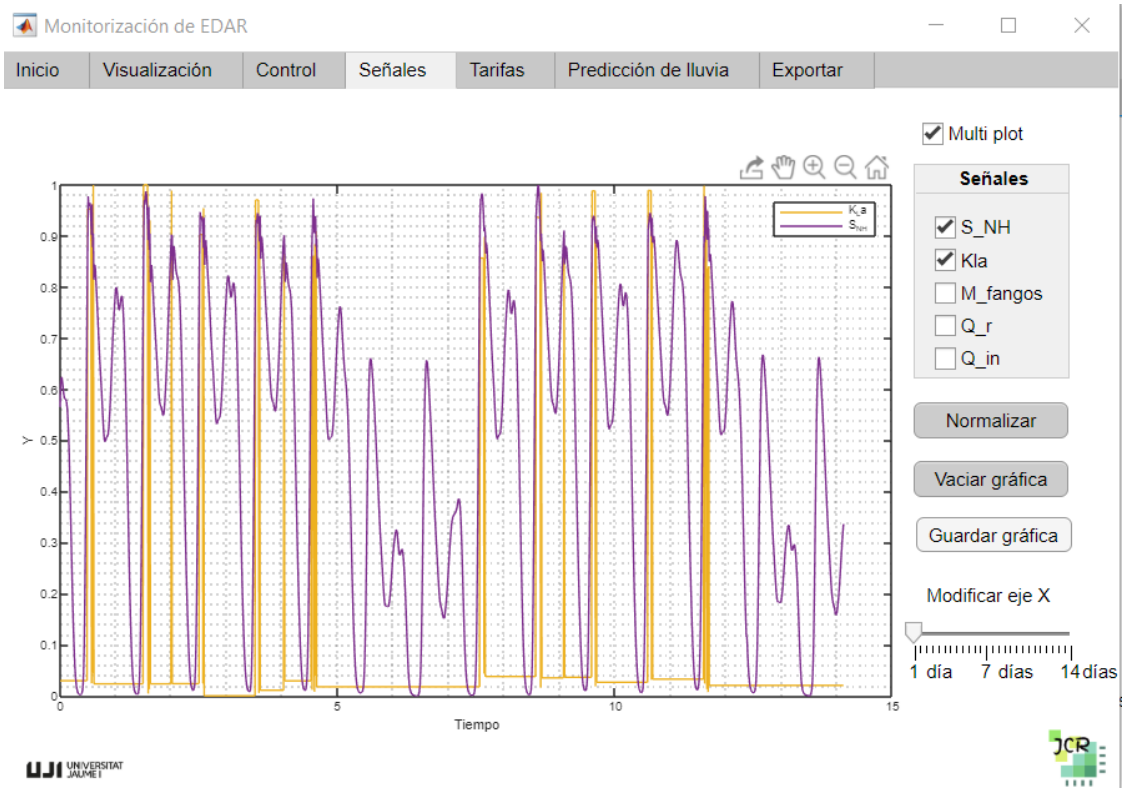


Figura 77.- Captura de pantalla de la pestaña de Señales/Histórico

Para poder modificar la cantidad de datos mostrados en la gráfica, se ha añadido un "slider" que hace variar el rango de visualización entre 1-14 días.

PESTAÑA 4 – *TARIFAS*

La siguiente pestaña hace referencia al avance del precio de la electricidad. En esta ventana, se pueden seleccionar las fechas de inicio y de fin de los datos a mostrar. De esta forma, al escoger un periodo válido y pulsar el botón de "Actualiza" se muestra la evolución del precio de la luz ente los días seleccionados, tanto en forma de gráfica como en forma de tabla.

De igual modo al mostrar los datos, aparece un mensaje en el que se indica el precio medio a lo largo de todo el periodo. Los datos mostrados se extraen de la API de la Red Eléctrica Española, conocida como E-SIOS (Sistema de Información del Operador del Sistema). De este modo, al pulsar el botón la aplicación realiza una petición a la base de datos de la API, y esta le devuelve los datos reclamados en formato JSON. La propia aplicación los interpreta y los muestra a través de la interfaz. De nuevo el botón de "Vaciar" se utiliza para resetear la pestaña.

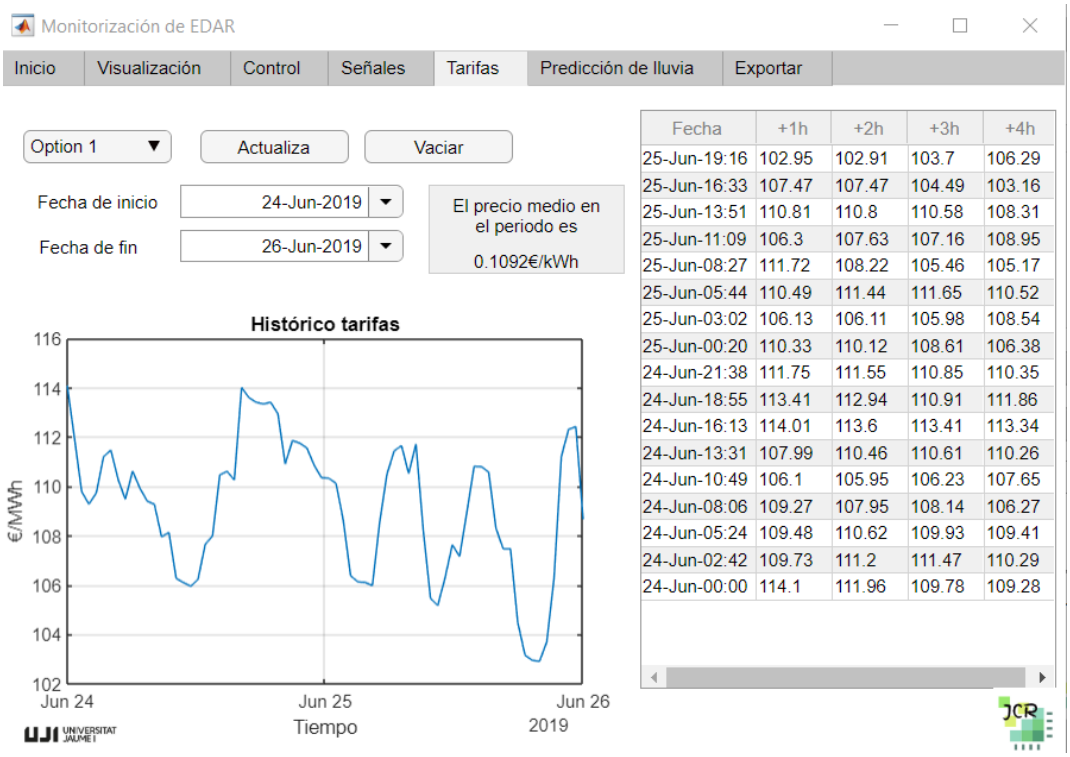


Figura 78.- Captura de pantalla de la pestaña de Tarifas eléctricas

PESTAÑA 5 - PREDICCIÓN DE LLUVIA

La pestaña de "Predicción de Lluvia" tiene como función mostrar el comportamiento de las precipitaciones en los próximos días y por consiguiente la predicción del caudal de entrada. Por el momento la actualización se realiza manualmente pulsando el botón "Actualiza", pero en una versión definitiva se podría realizar automáticamente transcurrido un cierto tiempo.

Al realizar la actualización, la aplicación se conecta a la API de "eltiempo.com", se generan una serie de tablas donde se recogen las predicciones de los próximos 5 días y se muestra una gráfica con los valores de lluvia esperados o el caudal que se prevé. Para seleccionar los datos mostrados en la gráfica se utiliza la lista desplegable en la que se puede elegir entre "Precipitaciones" o "Caudal". Como en los casos anteriores, el botón "Vaciar" resetea la pestaña.

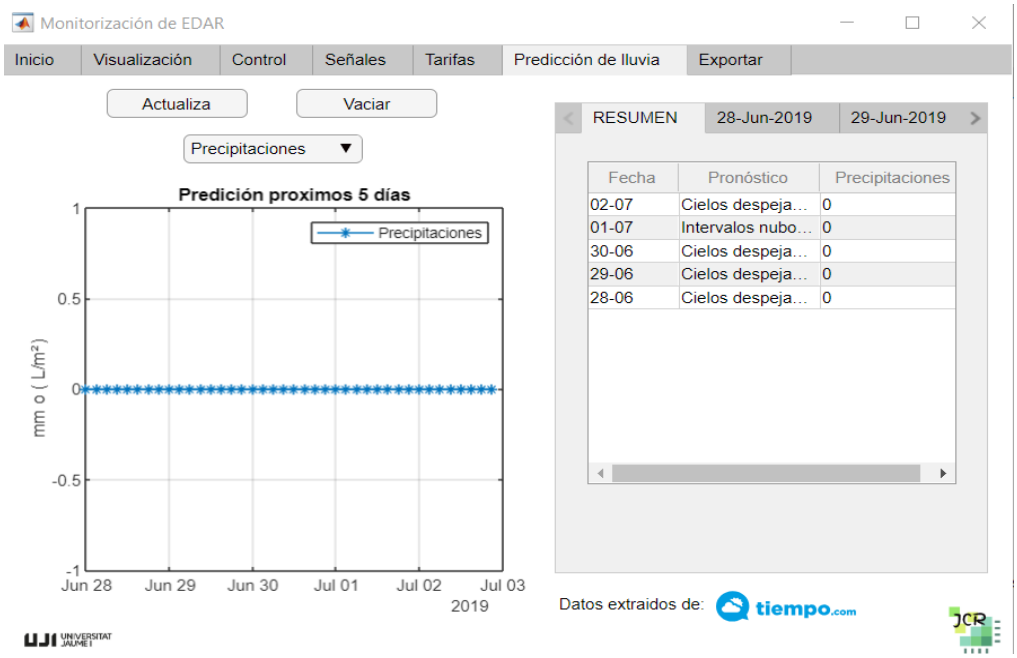


Figura 79.- Capturas de pantalla de la pestaña de predicción de lluvia

En la imagen anterior, a la derecha, se observa una ventana gris con una serie de pestañas. En la primera, con el nombre de "RESUMEN", aparecen en formato tabla tanto la descripción del tiempo para los próximos días, como el nivel de precipitaciones en L/m^2. Mientras que, en las siguientes se visualiza la predicción meteorológica de cada uno de los días siguientes. En la siguiente figura, se muestra el aspecto de una de estas pestañas de un día concreto.

Figura 80.- Tabla concreta para la predicción de un día

En estas pestañas específicas para cada fecha, se muestra un desglose de las predicciones cada 3 horas, empezando desde las 02:00 AM, de nuevo en descripción y en nivel de precipitaciones. Además, se ha incorporado un icono que permite conocer la predicción a simple viste y la misma información que aparece en la tabla resumen.

PESTAÑA 6 – EXPORTACIÓN DE DATOS

La última pestaña tiene como objetivo la extracción de datos, para ello se incorpora un listado con los diferentes datos que se han ido generando a lo largo de la utilización de la aplicación, así como un campo editable de texto que permite nombrar el archivo y seleccionar la carpeta de destino y el formato en el que se desean exportar los datos.

Esta pestaña es muy intuitiva, de hecho, los iconos para seleccionar el formato, así como los botones de exportación se muestran sombreados hasta que los datos de "selección de datos", "nombre de archivo" y "fichero de destino" no están correctamente cumplimentados. En la siguiente figura se muestra el aspecto de la pestaña cuando los datos introducidos en los campos son correctos y por consiguiente se puede proceder a la exportación.

Figura 81.- Capturas de pantalla de la pestaña de Extracción de datos

En el caso de los ficheros Excel, si entre una exportación y la siguiente no se modifican el nombre o la carpeta de destino, cada dato que se seleccione de la lista, se va incorporando a una nueva hoja del archivo con el nombre especificado. A continuación, se presenta un ejemplo.

	A	B	C	D	E	F	G
1	20/06/2019 8:53	Límite de Amonio superado					
2	20/06/2019 8:39	Límite de Amonio superado					
3	20/06/2019 8:24	Límite de Amonio superado					
4	20/06/2019 8:09	Límite de Amonio superado					
5	20/06/2019 7:54	Límite de Amonio superado					
6	20/06/2019 7:39	Límite de Amonio superado					
7	20/06/2019 7:24	Límite de Amonio superado					
8	20/06/2019 7:10	Límite de Amonio superado					
9	20/06/2019 6:55	Límite de Amonio superado					
10	20/06/2019 6:40	Límite de Amonio superado					
11	20/06/2019 6:25	Límite de Amonio superado					
12	20/06/2019 6:10	Límite de Amonio superado					
13	20/06/2019 5:55	Límite de Amonio superado					
14	20/06/2019 5:40	Límite de Amonio superado					
15	20/06/2019 5:26	Límite de Amonio superado					
16							
17							
18							
19							
20							
21							

Alarmas registradas | Valores diarios medios | Tarifas en periodo seleccionado | Predicción

Figura 82.- Archivo Excel con datos de exportación en diferentes hojas

I.8.7.- RESULTADO FINAL APLICACIÓN DISTRIBUIBLE

El empaquetado final es un ejecutable convencional que permite desplegar la aplicación mediante un archivo ".exe". Es decir, como cualquier otro software comercial. A continuación, se muestra una captura de pantalla de la carpeta del distribuible. En ella se observa tanto el ejecutable como los archivos gráficos de los iconos o el documento "Readme.txt" en el que se recogen los prerrequisitos del sistema, el listado de ficheros para despliegue y empaquetado de la aplicación.

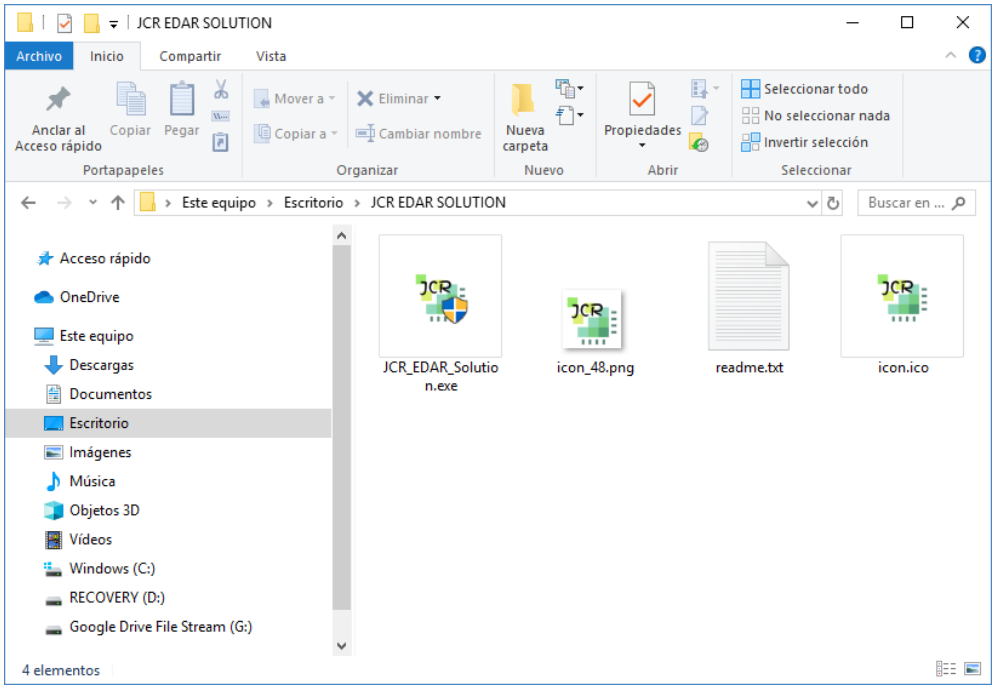


Figura 83.- Carpeta que contiene el ejecutable de la aplicación

Una vez se tiene acceso al ejecutable únicamente es necesario seguir los pasos recogidos en el **III.6.- MANUALES DE INSTALACIÓN DE APLICACIÓN DE CONTROL.**

I.9 PLANIFICACIÓN

El presente proyecto se ha dividido de la siguiente forma y la planificación del mismo se adaptará en función de las exigencias propias de su desarrollo:

- **FASE 1 – CONEXIÓN TORNADO MATLAB**
  - FASE 1.1. – DISEÑO DE PLANTA EN WEST
  - FASE 1.2. – EJECUCIÓN DESDE TORNADO
  - FASE 1.3. – LANZAMIENTO DE EXPERIMENTOS DESDE MATLAB
- **FASE 2 – CONSTRUCCIÓN DE SIMULADOR**
  - FASE 2.1. – SELECCIÓN DE VARIABLES DE ENTRADA Y SALIDA
  - FASE 2.2. – INTEGRACIÓN EN MATLAB
  - FASE 2.3. – VALIDACIÓN DEL SIMULADOR
- **FASE 3 – IDENTIFICACIÓN DE MODELO**
  - FASE 3.1. – EXCITACIÓN DEL SIMULADOR
  - FASE 3.2. – SELECCIÓN MÉTODO DE IDENTIFICACIÓN
  - FASE 3.3. – PRUEBA DE DIFERENTES ORDENES Y RETARDOS
  - FASE 3.4. – VALIDACIÓN Y EXPORTACIÓN DEL MODELO
- **FASE 4 – SELECCIÓN Y DESARROLLO DEL ALGORITMO DE CONTROL**
  - FASE 4.1. – ANÁLISIS DE ALTERNATIVAS
  - FASE 4.2. – DESARROLLO DEL MÉTODO SELECCIONADO
- **FASE 5 – INTEGRACIÓN DEL CONTROL**
  - FASE 5.1. – PREPARACIÓN DATOS Y PARÁMETROS
  - FASE 5.2. – DESARROLLO ALGORITMO DE CONTROL
  - FASE 5.3. – INCORPORACIÓN DEL CONTROL AL SIMULADOR
- **FASE 6 – DESARROLLO DE APLICACIÓN DE USUARIO**
  - FASE 6.1. – ANÁLISIS DE ALTERNATIVAS
  - FASE 6.2. – DISEÑO DE APLICACIÓN
- **FASE 7 – COMPILACIÓN Y ENCAPSULAMIENTO DE LA APLICACIÓN**
  - FASE 7.1. – SELECCIÓN FORMATO FINAL
  - FASE 7.2. – COMPILACIÓN
  - FASE 7.3. – VALIDACIÓN Y VERIFICACIÓN DE LA APLICACIÓN

Es importante destacar que a pesar de que la relación entre fases es de Fin-Fin (FF), al tratarse de un proyecto con gran parte de investigación las ramas irán la mayor parte del tiempo en paralelo.

I.10 ORDEN DE PRIORIDAD ENTRE LOS DOCUMENTOS

El orden de prioridad entre los documentos del presente proyecto se establece según la norma 157001:2014 “Criterios generales para la elaboración formal de los documentos que constituyen un proyecto técnico”.

I.11 ESTUDIO DE VIABILIDAD

La viabilidad del proyecto se ha dividido en dos partes para diferenciar su validez tecnológica y la repercusión económica de su implantación.

I.11.1.- VIABILIDAD TÉCNICA

A lo largo de este documento se ha justificado mediante argumentos objetivos que la ejecución de este proyecto es viable. Analizando los resultados obtenidos, se puede verificar que tanto las estrategias de control como la tecnología complementaría cumplen con los requisitos de diseño establecidos.

Puesto que los resultados obtenidos sobre el simulador son favorables y atractivos, se considera que la instalación en una EDAR real sería completamente posible e incluso recomendable.

Las pruebas de funcionamiento de la aplicación han mostrado unos resultados alentadores, tanto en velocidad de reacción como en grado de satisfacción de los usuarios encuestados. Para estos análisis de rendimiento se han seleccionado personas de diferentes edades, sexo y nivel académico con el fin de obtener puntos de vista variados.

I.11.2.- VIABILIDAD ECONÓMICA

Para validar la parte económica del proyecto, se ha realizado una comparativa entre el consumo de una planta controlada por un relé y con el control predictivo. A modo de aproximador del consumo energético se utiliza la siguiente expresión:

$$Energía_{día}[kWh] = \sum_{k=0}^{k=24*4} Potencia_{K_L a}[kW] * \frac{K_L a_k}{K_L a_{max}} * T_{activo}[h] \qquad (18)$$

Donde:

$\boldsymbol{K_L a_k}$ = El valor de la acción de control con un periodo de muestreo de 15 min.

$\boldsymbol{K_L a_{max}}$ = El valor máximo de la acción de control.

$\boldsymbol{Potencia_{K_L a}}$= Potencia nominal del soplador en kW.

$\boldsymbol{T_{activo}[h]}$ = Periodo durante el cual está la acción de control a un valor.

Por lo tanto, el ahorro se cuantifica de forma relativa, considerando la potencia nominal del soplador igual y se rige según la siguiente expresión.

$$Ahorro_{energético\ día}[\%] = 100 * \sum_{k=0}^{k=24*4} \frac{K_L a_{k_{MPC}}}{K_L a_{max}} \Big/ \sum_{k=0}^{k=24*4} \frac{K_L a_{k_{relé}}}{K_L a_{max}} \quad (19)$$

Donde:

$\boldsymbol{K_L a_{k_{MPC}}}$= Corresponde al valor instantáneo de K_La con el control MPC.

$\boldsymbol{K_L a_{k_{relé}}}$= Corresponde al valor instantáneo de K_La con el control tipo relé.

Extrapolando los resultados a términos económicos, la expresión del ahorro queda de la siguiente forma.

$$Ahorro_{econ.día}[\%] = 100 * \sum_{k=0}^{k=24*4} \frac{K_L a_{k_{MPC}} * precio_k}{K_L a_{max}} \Big/ \sum_{k=0}^{k=24*4} \frac{K_L a_{k_{relé}} * precio_k}{K_L a_{max}} \quad (20)$$

Utilizando las fórmulas anteriores y aproximando la potencia nominal del soplador a **100kW** se han obtenido los datos recogidos en la siguiente tabla.

	Relé	**MPC**
$\boldsymbol{Energía_{día}[kWh]}$	300 kWh/día	225 kWh/día
$\boldsymbol{Ahorro_{energético\ día}}[\%]$	Aprox. 25%	
$\boldsymbol{Ahorro_{econ.día}}[€/\boldsymbol{día}]$	Aprox. 12 €/día	

Tabla 16.- Tabla resumen de los resultados sobre viabilidad económica

Con estos datos y los recogidos en el **IV. PRESUPUESTO** se calcula el periodo de retorno del proyecto según la siguiente expresión.

$$PR[años] = \frac{Coste\ proyecto\ [€]}{365 * Ahorro\ económico_{diario}[€/día]} = \frac{23830\ [€]}{365\ [días] * 12\ [€/día]} \quad (21)$$

Por lo tanto, en aproximadamente **5 años y medio** (5.44 años) el proyecto empezará a generar beneficios para la planta en la que se haya instalado. Este valor es relativamente elevado ya que la parte de desarrollo de la aplicación está incluida en el coste total del proyecto, pero no se traduce en beneficio económico sino en **seguridad, precisión** y **comodidad**. Es decir, una parte considerable de este proyecto se dedicaría a evitar fallos humanos y mejorar el transcurso de la actividad diaria. Además, el ahorro energético se traduce en una **reducción de las emisiones de CO_2** de aproximadamente 7 toneladas de CO_2 equivalentes al año, considerando un factor de Mix energético de 0.27 kg de CO_2 / kWh (en el **ANEXO 4** se amplía la información sobre las tarifas y las emisiones de CO_2)

I.12 CONCLUSIÓN

En este apartado se recopilan tanto las conclusiones generales como los posibles trabajos futuros relacionados con el proyecto actual.

I.12.1.- CONCLUSIONES

A lo largo de este proyecto, se han presentado alternativas y diferentes caminos estudiados para cada una de las fases en las que se divide el mismo. Este análisis se ha realizado con el objetivo de obtener la solución óptima y más adaptada al caso concreto que en él se recoge.

Para empezar, uno de los resultados más satisfactorios es el hecho de haber diferenciado y separado completamente las fases de modelado, simulación y control de la planta. Esto permite la extrapolación y el escalado completo de esta aplicación a cualquier tipo de EDAR. De hecho, no se limitaría únicamente a Estaciones de Depuración, sino que se podría utilizar para simular y controlar cualquier tipo de proceso modelable en WEST (con un conjunto muy reducido de cambios en el método). Así mismo, en términos de eficiencia, al utilizar el *kernel* como motor de cálculo (en este caso Tornado, pero serviría cualquier *kernel* por el que se sustituyera) la velocidad es incluso mayor que la del propio software de modelado.

Durante la realización de pruebas se han detectado dos puntos clave de los modelos de predicción, por un lado, la importancia de una correcta excitación del sistema y por otro que los resultados finales de una estrategia MPC están condicionados por la validez de este modelo. En caso de comercializar la aplicación final de control uno de los puntos principales a tener en cuenta sería la formación de los operarios, en términos de identificación o actualización, y la validación del modelo.

En cuanto a la estrategia de control seleccionada, se han obtenidos datos alentadores. En ellos, se puede observar cómo las acciones de control se adelantan a los acontecimientos, modificando estas para conseguir llegar a las situaciones prefijadas. Además, mediante la implantación de este controlador, se consigue reducir tanto el consumo energético como el impacto económico del proceso. Por supuesto, esto se consigue manteniendo la salida de amonio por debajo de los límites legales y de diseño.

La parte visual de la aplicación de control da un gran valor añadido al proyecto, de hecho, hace el proceso de monitorización, seguimiento y mantenimiento del sistema mucho más intuitivo y cómodo, al igual que accesible a personas sin conocimientos en ingeniería de control o automatización. Su estructura de organización distribuida en pestañas, hace que la gestión de la misma sea mucho más simple y ordenada. De igual forma, el post-procesado de datos acoplado en la aplicación, generando ficheros y mostrando históricos, es una funcionalidad extra muy atractiva a la hora de su desarrollo industrial.

Finalmente, el método de compilación y distribución ha funcionado según lo previsto, ya que ha permitido desplegar la aplicación en ordenadores en los que no estaba instalado Matlab. Sin embargo, como se muestra a continuación, en las líneas futuras de trabajo se podría aumentar la eficiencia de la aplicación y su compilación, reduciendo de este modo su peso y su tiempo de despliegue.

I.12.2.- TRABAJO FUTURO

En base a las conclusiones y resultados obtenidos a lo largo del proyecto, se considera que las líneas por las que continuar investigando y trabajando son las siguientes:

- **APLICACIÓN SOBRE UNA EDAR REAL.** Si bien se ha realizado el proyecto pensando en su aplicación en cualquier tipo de EDAR, uno de los pasos a realizar es la interconexión con los sensores que recogen las medidas de las variables consideradas como medibles.
- **AUDITORÍA ENERGÉTICA Y SISTEMA DE GESTIÓN.** En temas de optimización de consumo energético es importante no realizar acciones aisladas, sino monitorizar los resultados de las modificaciones para verificar que los resultados son los esperado. También es necesario conocer los máximos consumidores de la planta para estudiar futuras acciones de ahorro económico.
- **RECURSIVIDAD DEL ALGORITMO.** En este tipo de control es interesante convertir el sistema en recursivo, para que se le realimente el error de predicción y de esta forma se puedan corregir las posibles desviaciones y se haga cada vez más preciso.
- **MEJORAS EN LAS TÉCNICAS DE IDENTIFICACIÓN SIMPLES.** La técnica actual de identificación tiene una fuerte componente empírica, ya que es el usuario el que defina las variables del modelo. Resultaría realmente valioso estudiar la forma de automatizar la construcción del modelo.
- **DETECCIÓN DE FALLOS.** Al tratarse de un sistema industrial es importante que sea capaz de prever los fallos y las tendencias negativas a la entrada del mismo. Por lo tanto, es interesante incorporar un algoritmo capaz de detectar estos casos.
- **COMBINACIÓN DE MODELOS DE IDENTIFICACIÓN.** Puesto que en el modelo se han detectado dos dinámicas con periodo de identificación muy diferenciados (la salida de amonio y el coste energético mensual), puede ser beneficioso el hecho de identificarlas por separado. Por un lado, crear un modelo que prediga los valores de salida de amonio con un periodo de tiempo lo más corto posible (aprox. 15min) y otro que lo haga con el coste energético con un periodo bastante superior (aprox. 2 días).
- **MACHINE LEARNING Y DEEP LEARNING.** El auge de la inteligencia artificial y los programas que aprenden cada vez que se ejecutan hace sospechar que es el camino a seguir en el futuro. De hecho, estos métodos permiten el tratamiento de una gran cantidad de datos, cosa que sería muy beneficioso si el control y la aplicación diseñadas en este proyecto se integran, en continuo, en una planta real.
- **OPTIMIZACIÓN DE LA APLICACIÓN.** La aplicación cumple con todos los requisitos impuestos, sin embargo, para hacer que sea más ligera y se despliegue más fácilmente podría ser útil reescribirla en otros lenguajes, como podrían ser Python o JavaScript.

II. ANEXOS

ÍNDICE DE CAPÍTULO

ÍNDICES DE TABLAS Y FIGURAS

ANEXO 1: EXTRACTO BSM1 (FÓRMULAS)

Para tener más información y poder una comprensión más profunda del sistema, a continuación, se presentan las fórmulas que rigen el comportamiento de la planta a simular.

LISTA DE PROCESOS

- j=1: *Aerobic growth of heterotrophs* - Crecimiento aeróbico de los heterótrofos

$$\rho_1 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{S_O}{K_{O,H} + S_O}\right) X_{B,H} \quad (I)$$

- j=2: *Anoxic growth of heterotrophs* -Crecimiento anóxico de los heterótrofos

$$\rho_2 = \mu_H \left(\frac{S_S}{K_S + S_S}\right)\left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\eta_g X_{B,H} \quad (II)$$

- j=3: *Aerobic growth of autotrophs* - Crecimiento aeróbico de los autótrofos

$$\rho_3 = \mu_A \left(\frac{S_{NH}}{K_{NH} + S_{NH}}\right)\left(\frac{S_O}{K_{O,A} + S_O}\right) X_{B,A} \quad (III)$$

- j=4: *Decay of heterotrophs* -Deterioro de los heterótrofos

$$\rho_4 = b_H X_{B,H} \quad (IV)$$

- j=5: *Decay of autotrophs* -Deterioro de los autótrofos

$$\rho_5 = b_A X_{B,A} \quad (V)$$

- j=6: *Ammonification of soluble organic nitrogen* -Amonificación del nitrógeno orgánico soluble

$$\rho_6 = k_a S_{ND} X_{B,H} \quad (VI)$$

- j=7: *Hydrolysis of entrapped organics* -Hidrólisis de la materia orgánica atrapada.

$$\rho_7 = k_h \frac{X_S/X_{B,H}}{K_X + (X_S/X_{B,H})}\left[\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_h \left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right] X_{B,H} \quad (VII)$$

- j=7: *Hydrolysis of entrapped organic nitrogen* -Hidrólisis del nitrógeno orgánico atrapado.

$$\rho_8 = k_h \frac{\frac{X_S}{X_{B,H}}}{K_X + \left(\frac{X_S}{X_{B,H}}\right)}\left[\left(\frac{S_O}{K_{O,H} + S_O}\right) + \eta_k \left(\frac{K_{O,H}}{K_{O,H} + S_O}\right)\left(\frac{S_{NO}}{K_{NO} + S_{NO}}\right)\right] X_{B,H}(X_{ND}/X_S) \quad (VIII)$$

DETALLES SOBRE EL LAYOUT DE LA PLANTA

BALANCES DE MASAS DE LOS REACTORES

- Para *k*=1 (*Unit1)*

$$\frac{dZ_{as,1}}{dt} = \frac{1}{V_{as,1}}\left(Q_{int}Z_{int} + Q_r Z_r + Q_i Z_i + r_{Z,1}V_{as,1} - Q_1 Z_{as,1}\right) \quad .(IX)$$

$$Q_1 = Q_{int} + Q_r + Q_i \quad .(X)$$

- Para *k*=2 a 5 (*Unit2- Unit5)*

$$\frac{dZ_{as,k}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}Z_{as,k-1} + r_{Z,k}V_{as,k} - Q_k Z_{as,k}\right) \quad .(XI)$$

$$Q_k = Q_{k-1} \quad .(XII)$$

- Caso especial para el oxígeno ($S_{O,as,k}$)

$$\frac{dS_{O,as,1}}{dt} = \frac{1}{V_{as,k}}\left(Q_{k-1}S_{O,as,k-1} + r_{Z,k}V_{as,k} + (K_L a)_k V_{as,k}\left(S_O^* - S_{O,as,k}\right) - Q_k S_{O,as,k}\right) \quad .(XIII)$$

Donde $S_O^* = 8g \cdot m^{-3}$ y $r_{Z,k}$ corresponde a la ratio de conversión apropiado.

- *Miscellaneous*

$$Z_{int} = Z_{as,5} \quad .(XIV)$$

$$Z_f = Z_{as,5} \quad .(XV)$$

$$Z_w = Z_r \quad .(XVI)$$

$$Q_f = \ Q_e + Q_r + Q_w = Q_e + Q_u \quad .(XVII)$$

SECONDARY CLARIFIER

	Parámetro	Unidades	Valor
Velocidad máxima de asentamiento	v_0'	$m.d^{-1}$	250
Velocidad Vesilind máxima de asentamiento	v_0	$m.d^{-1}$	474
Parámetro de zona impedida de asentamiento	r_h	$m^3.(gSS)^{-1}$	0.000576
Parámetro de zona de Floculante	r_p	$m^3.(gSS)^{-1}$	0.00286
Fracción no asentable	f_{ns}	[-]	0.00228

Tabla. I.- Parámetros de comportamiento del decantador

***Downward velocity**:* $v_{dn} = \frac{Q_u}{A} = Q_r + \frac{Q_w}{A}$.(XVIII)

Upwards velocity: $v_{up} = \frac{Q_e}{A}$.(XIX)

A modo de ejemplo se muestra la siguiente formula del balance de masas de fango, para consultar el resto el documento está disponible on-line. Para la capa de alimentación (*m*=6):

$$\frac{dX_{se,m}}{dt} = \frac{\frac{Q_f X_f}{A} + J_{se,m+1} - \left(v_{up} + v_{dn}\right)X_{sc,m} - \min\left(J_{s,m}, J_{s,m-1}\right)}{z_m} \quad .(XX)$$

ANEXO 2: MPC – MODEL based PREDICTIVE CONTROL

Actualmente, las investigaciones en el terreno de la ingeniería de control siguen una gran variedad de caminos. En la siguiente imagen se presenta un resumen de los principales tipos de controles ordenados en función de sus expectativas y posibilidades técnicas.

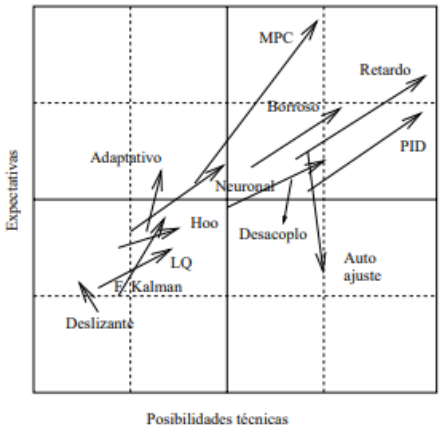


Figura. I.- Análisis de posibilidades técnicas de las diferentes estrategias de control
Fuente: I Curso de Especialización en Automática Aguadulce, Almería, 2000

En esta figura se observa que, en el campo del control de procesos industriales, el control predictivo Basado en Modelo tiene y va a continuar desarrollando una importancia a tener en cuenta, haciendo que se le pueda considerar una tecnología más que introducida en la industria.

Para entender el surgimiento del Control predictivo, es importante conocer que se desarrolló a partir de dos líneas básicas. Por un lado, aproximadamente en los últimos años del 1970 se empezaron a desarrollar algoritmos que predicen el resultado de aplicar las acciones de control en el futuro. Con esta predicción y buscando la minimización del error, se seleccionaban las acciones de control siguientes. La optimización se realizaba en cada instante del muestreo. Esta formulación era de naturaleza heurística y algorítmica que aprovechaba en la medida de lo posible el potencial de los computadores digitales del momento [12].

Además de esta creciente tendencia, fue surgiendo otra rama de trabajo enfocada al control adaptativo. El desarrollo se centró en procesos esencialmente monovariables formuladas con modelos entrada/salida.

Unificando las dos áreas de desarrollo e investigación nació el control predictivo generalizado (GPC), precursor de lo que ahora se denomina MPC o control predictivo basado en modelos.

Ya en 1997, Qin y Badgwel en uno de sus artículos enumeraban una gran cantidad de aplicaciones de este tipo de control, aproximadamente 2200, sobre todo en el sector petroquímico. Las razones principales del éxito del MPC a lo largo de su historia son las tres siguientes:

- El hecho de contar con un modelo explícito del proceso permite al controlador tener en cuenta todas las partes importantes de la dinámica del proceso.
- El concepto de horizonte temporal futuro permite pre-alimentar el efecto de las perturbaciones logrando que el controlador guie la salida a la trayectoria deseada.

- Las restricciones hacen que el controlador considere los criterios de diseño evitando que sobrepase ciertos valores y aumentando la precisión del mismo.

ESTRUCTURA BÁSICA DE UN CONTROL PREDICTIVO POR MODELO (MPC)

Una vez presentado el funcionamiento y el origen de la estrategia MPC, se presenta su estructura básica, en la siguiente figura, se muestra un esquema grafico de las partes con las que cuenta cualquier control de este tipo.

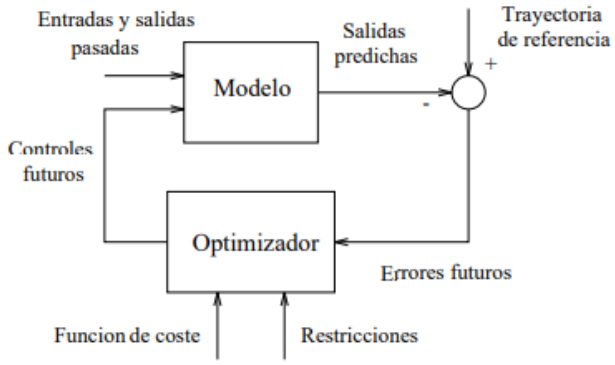


Figura. II.- Estructura básica de un MPC

La definición de cada una de estas partes se presenta a continuación:

- **Modelo:** Esta es una de las partes fundamentales y de mayor importancia del sistema de control, su función principal es la de replicar y predecir el comportamiento del proceso real. De esta forma se puede comprobar cuál será el resultado de aplicar las diferentes acciones de control y elegir la más adecuada. Las entradas y salidas de este bloque son las siguientes:
    - ***Entradas y salidas pasadas***. – Son entrada de este bloque. Este tipo de modelos se basa en los datos anteriores para conocer los valores del futuro.
    - ***Controles futuros.*** – Son salida del bloque de *Optimización* y constituyen las variables de decisión del modelo.
    - ***Salidas predichas.*** - Son la salida de este bloque. Son las salidas que este bloque predice hasta el horizonte temporal.
- **Comparador / cálculo del error:** En este punto se realiza la comparación entre la salida predicha y la trayectoria de referencia. Esta parte del esquema es simplemente una diferencia entre las señales de entrada.
    - ***Trayectoria de referencia*****. -** Es entrada del bloque y representa la trayectoria que se desea para la salida del proceso.
    - ***Salidas predichas*****. –** Son entrada de este módulo y la salida del modelo, sirven para conocer el error entre el valor deseado y el que se obtendrá realmente.
    - ***Errores futuros*****. –** Es salida de esta parte y representa el grado de seguimiento que se está realizando a la trayectoria de referencia.
- **Optimizador:** Este es uno de los bloques más importantes y consiste en una serie de algoritmos matemáticos, que se utiliza para llegar a un punto óptimo de funcionamiento.

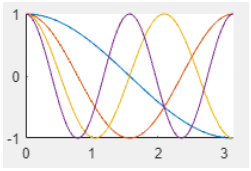
- ***Función de coste***. – La función de coste o función objetivo es aquella que se define para minimizar. Es decir, es una fórmula que relaciona las variables decisión, de forma que se optimiza su valor final.
- ***Restricciones***. – Las restricciones son los valores límite que puede utilizar el optimizador. De esta forma se acota el comportamiento del optimizador y se guía para obtener con la menor carga de cálculo posible.
- ***Errores futuros***. – Los errores futuros son la comparación de las señales predichas con la trayectoria de referencia.
- ***Controles futuros*. –** Los controles futuros son la salida del optimizador y corresponde con los valores que minimizan la función objetivo. Estos controles pasan al modelo para comprobar su efecto.

ANEXO 3: LIBRERÍA APP DESIGNER

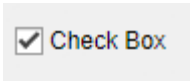
GALERÍA DE COMPONENTES

COMMON COMPONENTS / Componentes estándar

AXES

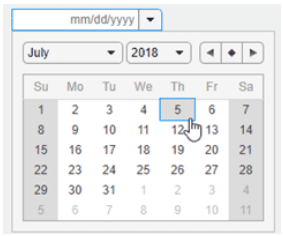


CHECK BOX

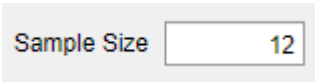


SPINNER

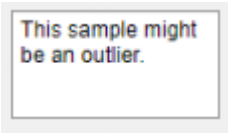
DATA PICKER



EDIT FIELD (NUMBER)



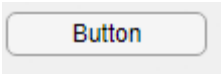
TEXT AREA



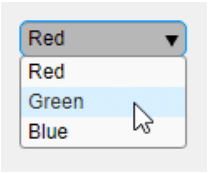
EDIT FIELD (TEXT)

STATE BUTTON

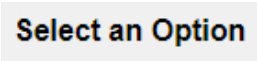
BUTTON



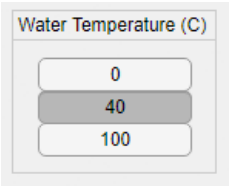
DROP DOWN



LABEL

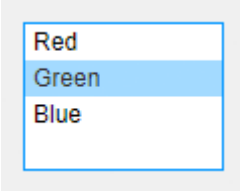


TOGGLE BUTTON GROUP



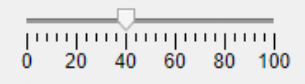
IMAGE

LIST BOX

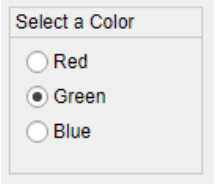


TABLE

SLIDER



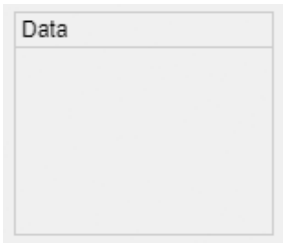
RADIO BUTTON GROUP



TREE

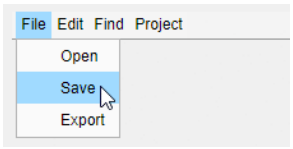
CONTAINERS AND FIGURE TOOLS/ Contenedores y herramientas de figura

PANEL



TAB GROUP

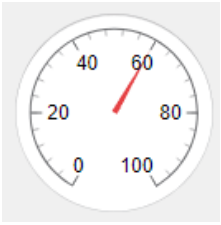
MENU BAR



GRID LAYOUT

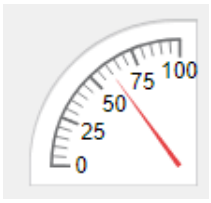
INSTRUMENTATION / Instrumentación

GAUGE



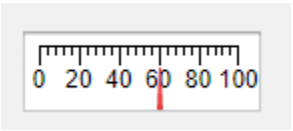
SWITCH

90 DEGREE GAUGE

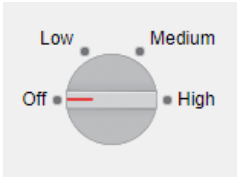


TOGGLE SWITCH

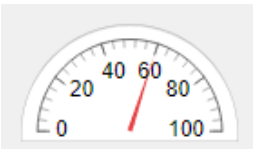
LINEAR GAUGE



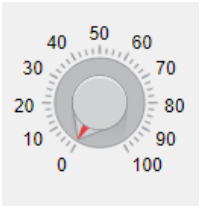
DISCRETE KNOB



SEMICIRCULAR GAUGE



KNOB



LAMP

ROCKER SWITCH

ANEXO 4: CAUDALES FUTUROS, TARIFA ELÉCTRICA Y MIX ENERGÉTICO

Una de las cuestiones más importantes de este tipo de controles es la predicción de cada una de las variables, ya que en función de estas predicciones se podrá considerar que la salida será de mayor o menor calidad y precisión. Para comprobar las predicciones, se debe prestar especial atención al precio de la electricidad y al caudal de entrada.

Para poder realizar una buena estimación de ambas variables es importante conocer que dependen del:

- **CAUDAL DE ENTRADA: e**l caudal de entrada de una EDAR depende del comportamiento de la ciudad y de las aguas recogidas en las zonas de recolección. Por otro lado, uno de los factores que más afectan a esta variable son las condiciones meteorológicas. Es decir, cuando se produce una tormenta el caudal aumenta repentinamente llegando incluso a sobrepasar el máximo admisible, por el contrario, si se trata de lluvias moderadas aumenta el caudal, pero de una forma más suave y estable.
- **TARIFA ELÉCTRICA:** El precio final de la electricidad, depende de la tarifa que se tenga contratada y de si se cuenta con discriminación horaria o no.

PREDICCIÓN DE CAUDAL DE ENTRADA

Para poder estimar o predecir esta variable se ha realizado un análisis de los datos que ponen a disposición en los ficheros del BSM1:

- **"Inf_dry_2006.txt"** – En este fichero se recogen los datos de 14 días sin precipitaciones, en los que se observa perfectamente el comportamiento habitual de esta variable.
- **"Inf_rain_2006.txt"** – En este fichero se recogen los datos de 14 días en los que en dos de ellos se han producido precipitaciones moderadas y con cierta estabilidad.
- **"Inf_strm_2006.txt"** – En este fichero se recogen los datos de 14 días en los que en dos momentos puntuales se producen tormentas, haciendo que los valores se disparen de una forma un tanto descontrolada y acelerada.

Para poder comparar a simple vista los ficheros se han graficado en la siguiente figura:

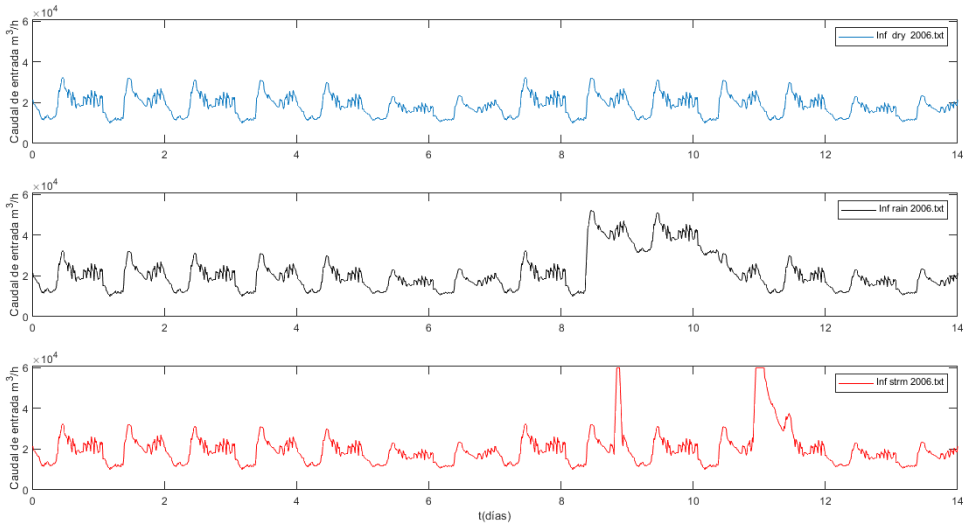


Figura. III.- Comparación de los diferentes ficheros de caudal del BSM1

Se observa que la mayor parte del tiempo los caudales son idénticos, haciéndose diferentes cuando se producen los fenómenos meteorológicos antes citados. A partir de estos datos, se puede afirmar que para que la predicción de esta variable se realice correctamente se debe tener en cuenta el día de la semana en el que se empieza a simular, ya que el patrón es diferente entre semana y en el fin de semana, y cuál es la predicción meteorológica para los próximos días.

Con el fin de hacer esto de una forma sistemática y sencilla, se ha decidido acceder desde el propio programa de control mediante una API, a los datos proporcionados por una de las agencias de meteorología. Una vez conocida la predicción de las precipitaciones se realiza una estimación del caudal considerando los datos conocidos. Es decir, conocer los datos de precipitaciones de los próximos días nos ayuda a conocer de antemano el comportamiento del caudal entrante. Además, para que la simulación sea más real, se irán modificando de forma aleatoria los perfiles del patrón de caudal para que no sean en todo momento iguales. En la siguiente figura se presenta la comparación de un día soleado, uno lluvioso y uno de tormenta. Estos son los datos que servirán para estimar el caudal cuando en función de la predicción meteorológica.

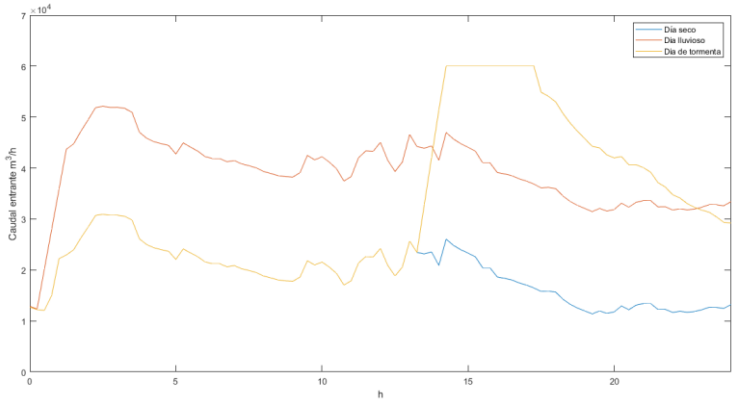


Figura. IV.- Comparación entre los días secos y de lluvia

Una vez extraído el valor o rango de valores en los que se encuentra el caudal entrante en función del nivel de precipitaciones se establece un protocolo que genera un fichero de entrada basándose en las predicciones recogidas de la API. A continuación, se presenta una gráfica a modo de ejemplo, en ella se ve como se modifica el caudal de entrada en función del nivel de precipitaciones.

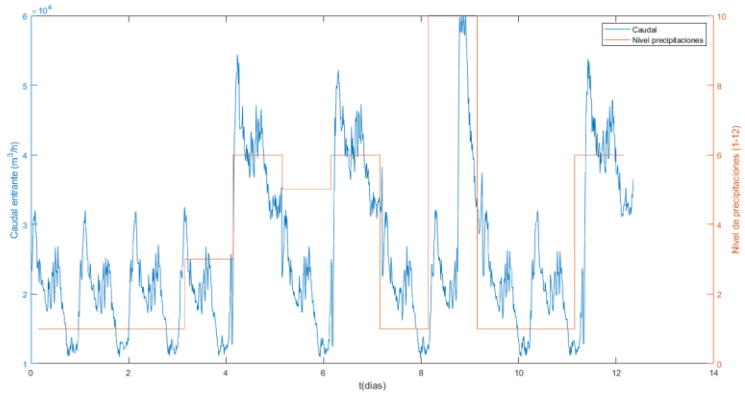


Figura. V.- Relación entre nivel de precipitaciones y caudal de entrada

La escala del nivel de precipitaciones se basa en la que utiliza la API seleccionada, y se muestra a continuación:

Símbolo	Descripción del símbolo
1	Cielos despejados
2	Intervalos nubosos
3	Cielos nubosos
4	Cielos cubiertos
5	Intervalos nubosos con lluvias débiles
6	Cielos nubosos con lluvias débiles
7	Cielos cubiertos con lluvias débiles
8	Intervalos nubosos con lluvias moderadas
9	Cielos nubosos con lluvias moderadas
10	Cielos cubiertos con lluvias moderadas
11	Intervalos nubosos con chubascos tormentosos
12	Cielos cubiertos con chubascos tormentosos

Tabla. II.- Predicciones y descripciones del clima

Es importante destacar que estos datos se pueden utilizar para prever el comportamiento de la EDAR pero no para el control. Es decir, se usan en la aplicación a fin de mantener informados a los usuarios de posibles cambios, pero no los tiene en cuenta el modelo. De hecho, para estas fluctuaciones no previstas existen las técnicas de detección de fallos.

PREDICCIÓN DE LA TARIFA ELÉCTRICA

Entorno a este tema hay una gran cantidad de literatura, es un tema recurrente en muchos grupos de investigación. Muchas de estas ramas pretenden realizar predicción del coste de la electricidad en función de los datos de radiación, vientos y demás fenómenos meteorológicos. Sin embargo, en este proyecto se usará una **TARIFA DE ACCESO A REDES ELÉCTRICAS 3.X CON SEIS PERÍODOS**. En la siguiente tabla se recogen los datos sobre esta tarifa y la temporalización de los diferentes periodos**.** La **X** depende de la potencia total instalada. Para este proyecto se utilizan las tarifas de los meses invernales.

HORAS DE	HORAS A	ENERO	FEBRERO	MARZO	ABRIL	MAYO	JUNIO	JULIO	AGOSTO	SEPTIEMBRE	OCTUBRE	NOVIEMBRE	DICIEMBRE
00	01	6	6	6	6	6	6	6	6	6	6	6	6
01	02	6	6	6	6	6	6	6	6	6	6	6	6
02	03	6	6	6	6	6	6	6	6	6	6	6	6
03	04	6	6	6	6	6	6	6	6	6	6	6	6
04	05	6	6	6	6	6	6	6	6	6	6	6	6
05	06	6	6	6	6	6	6	6	6	6	6	6	6
06	07	6	6	6	6	6	6	6	6	6	6	6	6
07	08	6	6	6	6	6	6	6	6	6	6	6	6
08	09	2	2	4	5	5	4	2	2	6	4	5	4	2
09	10	2	2	4	5	5	3	2	2	6	3	5	4	2
10	11	1	1	4	5	5	3	2	2	6	3	5	4	1
11	12	1	1	4	5	5	3	1	1	6	3	5	4	1
12	13	1	1	4	5	5	3	1	1	6	3	5	4	1
13	14	2	2	4	5	5	3	1	1	6	3	5	4	2
14	15	2	2	4	5	5	3	1	1	6	3	5	4	2
15	16	2	2	4	5	5	4	1	1	6	4	5	4	2
16	17	2	2	3	5	5	4	1	1	6	4	5	3	2
17	18	2	2	3	5	5	4	1	1	6	4	5	3	2
18	19	1	1	3	5	5	4	1	1	6	4	5	3	1
19	20	1	1	3	5	5	4	2	2	6	4	5	3	1
20	21	1	1	3	5	5	4	2	2	6	4	5	3	1
21	22	2	2	3	5	5	4	2	2	6	4	5	3	2
22	23	2	2	4	5	5	4	2	2	6	4	5	4	2
23	00	2	2	4	5	5	4	2	2	6	4	5	4	2
SÁBADOS		6	6	6	6	6	6	6	6	6	6	6	6
DOMINGOS Y FESTIVOS*		6	6	6	6	6	6	6	6	6	6	6	6
		6	6	6	6	6	6	6	6	6	6	6	6

DE LUNES A VIERNES

Tabla. III.- TARIFA 6.X

Ya que estos datos se deben introducir en el código del algoritmo de control, se ha desarrollado una función que calcula el precio del kWh en función de la hora del día. Los resultados para un día completo de los meses centrales del año son los siguientes.

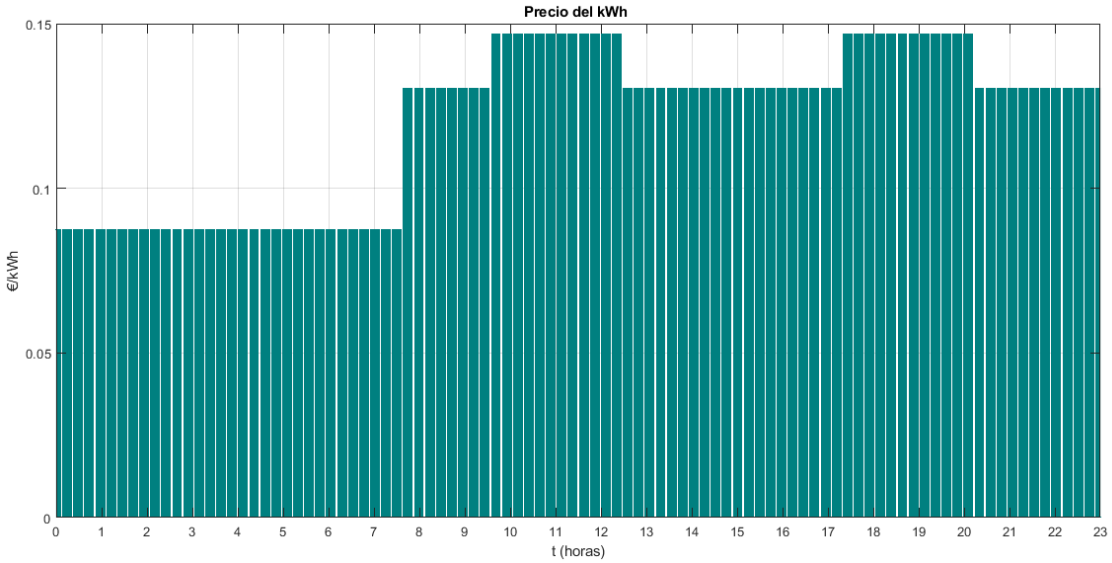


Tabla. IV.- Evolución del precio del kWh

Estos precios son los que usa el optimizador para calcular cuando es el mejor momento para activar las acciones de control minimizando el consumo. Aunque en la aplicación no solo se muestran los datos de la tarifa, sino que se puede seleccionar varias fechas y hacer informes históricos de precios. De hecho, en la aplicación se puede seleccionar la tarifa que se quiere mostrar para así poder comparar y valorar un posible cambio.

FACTOR DE MIX ENERGÉTICO

En el documento "**REGISTRO DE HUELLA DE CARBONO, COMPENSACIÓN Y PROYECTOS DE ABSORCIÓN DE DIÓXIDO DE CARBONO**", publicado en abril de 2019 por el Ministerio para la Transición Ecológica del Gobierno de España, se encuentran los factores de emisión de cada una de las comercializadoras que se han de utilizar para el correcto cálculo de la huella de carbono.

Además, se presenta la fórmula para realizar dicho cálculo y es la siguiente:

$$Emisiones_{CO_2}[kg\ CO_2] = Energía\ cons\ [kWh] \cdot factor\ emisión\ \left[\frac{kg\ de\ CO_2 eq}{kWh}\right] \qquad .(XXI)$$

ANEXO 5: CÓDIGO MATLAB

A5.1.- MATRICES PARA MODELO

```
%Carga modelo
load('modelo.mat')
n=modelo.n;
theta=modelo.theta;

%Extracción de las thetas para cada variable
t_y1=theta(2:(n(1)+1));
t_u1=theta(n(1)+2:(n(1)+n(2)+1));
t_u2=theta((n(1)+n(2)+2):(n(1)+n(2)+n(3)+1));
t_wQ_1=theta((n(1)+n(2)+n(3)+2):(n(1)+n(2)+n(3)+n(4)+1));
t_wQ_2=theta(n(1)+n(2)+n(3)+n(4)+2:n(1)+n(2)+n(3)+2*n(4)+1);
t_wQ_3=theta(n(1)+n(2)+n(3)+2*n(4)+2:n(1)+n(2)+n(3)+3*n(4)+1);

% Relleno de matriz M o A
M(1,1:n(1))=t_y1;
M(2:n(1),1:n(1)-1)=eye(n(1)-1);

M(1,n(1)+1:n(1)+n(2)-1)=t_u1(2:end)';
M(n(1)+2:n(1)+n(2)-1,n(1)+1:n(1)+n(2)-2)=eye(n(2)-2);

M(1,n(1)+n(2):(n(1)+n(2)+n(3)-2))=t_u2(2:end)';
M(n(1)+n(2)+1:n(1)+n(2)+n(3)-2,n(1)+n(2):n(1)+n(2)+n(3)-3)=eye(n(3)-
2);

% Relleno de matriz B
[filas columnas]=size(M);
B=zeros(filas,2);
B(1,1)=t_u1(1);
B(n(1)+2,1)=1;
B(1,2)=t_u2(1);
B(n(1)+2+n(2),2)=1;

% Relleno de matriz C
C(1,1)=theta(1);
C=[C; zeros(filas-1,1)];

% Relleno de matriz Q
Q=[t_wQ_1' t_wQ_2' t_wQ_3'];
Q=[Q ; zeros(filas-1,length(Q))];
```

A5.2.- FUNCIÓN PARA NORMALIZAR

```
function y=normalizar(x,maximo,minimo,modo)
      if modo==1
          y=(x-minimo)/(maximo-minimo);
      else
          y=x*(maximo-minimo)+minimo;
      end
```

A5.3.- FUNCIÓN PARA SIMULACIÓN WEST

```
function [y,ok]=SimulacionWest(TimeStop , MaxTimeStep, Kla, Q_Out2 ,
New, Times , Name )
%Carpeta en la que se genera y se guarda el experimento
data='C:\Users\JCR\Desktop\SIMULACIONES\';
NewDir=strcat(data,Name);
DynamicObjEval=strcat(data,Name,'\BSM1_OL_ASM1.Dynamic.ObjEval.Exp.xml');
Municipality=strcat(data,Name,'\Influent_Municipality_1.Dynamic.in.txt');
% Condición que se cumple cuando el archivo es nuevo y crea la nueva
% carpeta
 if New==1
    mkdir(NewDir);
    copyfile('C:\Users\JCR\Desktop\SIMULACIONES\BSM1_JCR',NewDir);
    modificar_Tiempo_Outputs(0,DynamicObjEval)
    modificar_XML(TimeStop,MaxTimeStep,DynamicObjEval);
end

ok=1;
intentos=0;

%Bucle para repetir el experimento
for i=1:Times
    ok=1;
    [status, cmdout]=Tornado_TEXEC( Kla, Q_Out2,DynamicObjEval);
    if status==1
        ok=0;
        disp('Error en tornado')
        disp(cmdout)
        % Envío de correo en caso de fallo
        send_mail_message('jcandearom','MATLAB','Error tornado ','123.mat')
        while (status==1 && intentos<10)
            [status, cmdout]=Tornado_TEXEC(Kla, Q_Out2,DynamicObjEval);
            intentos=intentos+1;
        end
        disp(['Error en tornado solucionado ', num2str(intentos)])
        y=SaveOutputs(DynamicObjEval);
        break
    end
    y=SaveOutputs(DynamicObjEval);

    % Hacer avanzar el fichero de entrada
    reescribir(Municipality,TimeStop);
end
```

A5.4.- FUNCIÓN PARA SIMULACIÓN WEST

```
function [y1 , y2, u1, u2, wQ]=SimuladorWWTP(u1,u2,New,Name)
Kla=u1;
Q_Out2=u2;
TimeStop=0.0104;
MaxTimeStep=TimeStop/100;
Times=1;
% Lanzamiento de la simulación
[y,ok]=SimulacionWest(TimeStop,MaxTimeStep,Kla,Q_Out2,New,Times,Name);

if ok==1
    %Lectura de variables de salida
    Kla=y.ASU5(end,44)';
    S_nh=y.Out_1(end,24)';
```

```
    t=y.ASU5(end,1)';
    wQ=y.Muni(end,2)';
    Q_out2=y.FS_1(end,51)';
    M_fangos=sum(y.Waste(end,end-14:1:end-1)');
    %Renombrar las variables
    y1=S_nh;
    y2=M_fangos;
    u1=Kla;
    u2=Q_out2;
else
    %Lectura de variables de salida
    Kla=y.ASU5(end-100,44)';
    S_nh=y.Out_1(end-100,24)';
    t=y.ASU5(end-100,1)';
    wQ=y.Muni(end-100,2)';
    Q_out2=y.FS_1(end-100,51)';
    M_fangos=sum(y.Waste(end-100,end-14:1:end-1)');
    %Renombrar las variables
    y1=S_nh;
    y2=M_fangos;
    u1=Kla;
    u2=Q_out2;
end
```

A5.5.- FUNCIÓN PARA EJECUTAR TORNADO

```
function [status, cmdout]=Tornado_TEXEC( Kla, Q_Out2,fileName)
    A='texec -sd -cp ';
    A1=fileName;
    A2=' -i ".ASU_5.Kla=';
    B=num2str(Kla);
    B1=' ';
    C=';.FS_1.Q_Out2=';
    D=num2str(Q_Out2);
    E='" ';
    F=fileName;
    %System se utiliza para ejecutar comandos en el icono del sistema
    [status, cmdout]=system([A A1 strcat(A2,B) strcat(C,D) E F] );
```

A5.6.- FUNCIÓN PARA MODIFICAR XML CARACTERÍSTICAS DEL EXPERIMENTO

```
function modificar_XML(a,b,fileName)
      % <Time>
      %     <Props>
      %         <Prop Name="StartTime" Value="0"/>
      %         <Prop Name="StopTime" Value="30"/>
      %         <Prop Name="Times" Value=""/>
      %         <Prop Name="EnableRealTime" Value="false"/>
      %         <Prop Name="Factor" Value="1"/>
      %     </Props>
      % </Time>

      StopTime=num2str(a);
      xDoc=xmlread(fullfile(fileName));
      allListItems=xDoc.getElementsByTagName('Time');
      allListItems=allListItems.item(0);
      thisItems1=allListItems.getElementsByTagName('Props');
      thisItems1=thisItems1.item(0);
      thisItems2=thisItems1.getElementsByTagName('Prop');
      thisItems2=thisItems2.item(1);
      thisItems2.setAttribute('Value',StopTime);

      % <Solve>
      % <Integ Method="RK4ASC">
      %   <Props>
      %       <Prop Name="MaxNoSteps" Value="0"/>
      %       <Prop Name="InitialStepSize" Value="1e-005"/>
      %       <Prop Name="MinStepSize" Value="1e-005"/>
      %       <Prop Name="MaxStepSize" Value="1"/>
      %       <Prop Name="Accuracy" Value="1e-008"/>
      %       <Prop Name="CheckMinStepSize" Value="false"/>
      %   </Props>
      % </Integ>
      MaxStepSize=num2str(b);
      allListItems2=xDoc.getElementsByTagName('Solve');
      allListItems2=allListItems2.item(0);
      thisItems2=allListItems2.getElementsByTagName('Integ');
      thisItems2=thisItems2.item(0);
      thisItems3=thisItems2.getElementsByTagName('Props');
      thisItems3=thisItems3.item(0);
      thisItems4=thisItems3.getElementsByTagName('Prop');
      thisItems4=thisItems4.item(1);
      thisItems4.setAttribute('Value',MaxStepSize);

      xmlwrite(fileName,xDoc);
      xmlremoveextralines(fileName);
```

A5.7.- FUNCIÓN PARA MODIFICAR XML CAMBIAR TIEMPO

```
function modificar_Tiempo_Outputs(a,fileName)

CommInt=num2str(a);
xDoc=xmlread(fullfile(fileName));

for i=1:12
    allListItems=xDoc.getElementsByTagName('File');
    allListItems=allListItems.item(i);
    thisItems1=allListItems.getElementsByTagName('Props');
    thisItems1=thisItems1.item(0);
```

```
    thisItems2=thisItems1.getElementsByTagName('Prop');
    thisItems2=thisItems2.item(2);
    thisItems2.setAttribute('Value',CommInt);
end

xmlwrite(fileName,xDoc);
xmlremoveextralines(fileName);
```

A5.8.- FUNCIÓN PARA GUARDAR LAS VARIABLES DE SALIDA

```
%SAVE OUTPUTS
function y=SaveOutputs(fileName)
folder=fileparts(fileName);
doc=char('ASU1','ASU2','ASU3','ASU4','ASU5','Clarifier','Corss','FS_1'
,'FS_2','Muni','Out_1','Waste');
file=strcat(folder,'\Outputs.mat');
    if exist(file,'file')
        load(file);
    else
        Out=[];
    end

    if isstruct(Out)
        a=fieldnames(Out);
        a=char(a(1));
    else
        a=1;
    end

if ~isempty(Out)
    saltot=Out.(a)(end,1)+Out.(a)(2,1)-Out.(a)(1,1);
end
for i=1:12
    dock=strrep(doc(i,:),' ','');
    if length(dock)==length('ASU5')
        if dock=='ASU5'

A=dlmread(strcat(folder,'\',dock,'.Dynamic.Simul.out.txt'),'\t',2,0);
        else

A=dlmread(strcat(folder,'\',dock,'.Dynamic.Simul.1.out.txt'),'\t',2,0);
        end
    else

A=dlmread(strcat(folder,'\',dock,'.Dynamic.Simul.1.out.txt'),'\t',2,0);
    end

    if isfield(Out,dock)
       A(:,1)=A(:,1)+saltot;
       Out.(dock)=[Out.(dock) ; A];
    else
        Out.(dock)=A;
    end
    A=[];
end
    y=Out;
    save(file)
end
```

A5.9.- FUNCIÓN PARA ENVIAR MAILS

```
function send_mail_message(id,subject,message,attachment)
%% SEND_MAIL_MESSAGE send email to gmail once calculation is done
% Example
%  send_mail_message('its.neeraj','Simulation  finished','This  is  the
message area','results.doc')

% Pradyumna
% June 2008
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Your gmail ID and password
%(from which email ID you would like to send the mail)
mail = 'notificacionesmatlabtfm@gmail.com';    %Your GMail email address
password = '*******';           %Your GMail password
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if nargin == 1
    message = subject;
    subject = '';
elseif nargin == 2
    message = '';
    attachment = '';
elseif nargin == 3
    attachment = '';
end
% Send Mail ID
emailto = strcat(id,'@gmail.com');
%% Set up Gmail SMTP service.
% Then this code will set up the preferences properly:
      setpref('Internet','E_mail',mail);
      setpref('Internet','SMTP_Server','smtp.gmail.com');
      setpref('Internet','SMTP_Username',mail);
      setpref('Internet','SMTP_Password',password);
% Gmail server.
      props = java.lang.System.getProperties;
      props.setProperty('mail.smtp.auth','true');
      props.setProperty('mail.smtp.socketFactory.class',
      'javax.net.ssl.SSLSocketFactory');
      props.setProperty('mail.smtp.socketFactory.port','465');
%% Send the email
if strcmp(mail,'GmailId@gmail.com')
    disp('Please provide your own gmail.')
    disp('You can do that by modifying the first two lines of the code')
    disp('after the comments.')
end
sendmail(emailto,subject,message,attachment)
end
```

A5.10.- FUNCIÓN PARA REESCRIBIR EL FICHERO DE ENTRADA

```
%FUNCIÓN  QUE  PERMITE  MODIFICAR  EL  ARCHIVO  fileName  (de  estructura
%fija)PARTIENDO DE LA ULTIMO VALOR UTILIZADO
function reescribir(fileName,ultimo_a)
      A=dlmread(fileName,'\t',2,0);
      t=A(:,1)';
      ultimo_a= find(t>ultimo_a);
      ultimo_a=ultimo_a(1);
      a=2;
% t=reordenar(A(:,a),ultimo_a);a=a+1;
      H2O=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_ALK=reordenar(A(:,a)',ultimo_a); a=a+1;
```

```
      S_I=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_ND=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_NH=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_NO=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_O=reordenar(A(:,a)',ultimo_a); a=a+1;
      S_S=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_BA=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_BH=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_I=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_ND=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_P=reordenar(A(:,a)',ultimo_a); a=a+1;
      X_S=reordenar(A(:,a)',ultimo_a);

      MAT = [t ;H2O; S_ALK; S_I; S_ND; S_NH; S_NO; S_O; S_S; X_BA;...
      X_BH; X_I; X_ND; X_P; X_S];
y=[MAT'];
fileID = fopen(fileName,'w');
fprintf(fileID,'\x23.%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%
s\t%s\t%s
\r\n','t','H2O','S_ALK','S_I','S_ND','S_NH','S_NO','S_O','S_S','X_BA',
'X_BH','X_I','X_ND','X_P','X_S');
fprintf(fileID,'\x23%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s\t%s
\t%s\t%s                                                                ...
\r\n','d','m3/d','g/m3','g/m3','g/m3','g/m3','g/m3','g/m3','g/m3','g/m
3','g/m3','g/m3','g/m3','g/m3','g/m3');
fprintf(fileID,'%.5f\t%.0f\t%.0f\t%.0f\t%.6f\t%.5f\t%.3f\t%.3f\t%.5f\t
%.3f\t%.3f\t%.3f\t%.3f\t%.3f\t%.3f \r\n',MAT);
fclose(fileID);
```

A5.11.- FUNCIÓN PARA REORDENAR VECTORES

```
function y=reordenar(b,ultimo_a)
      b_1=b(ultimo_a:end);
      b_2=b(1:ultimo_a-1);
      y=[b_1 b_2];
```

A5.12.- CÓDIGO DE SIMULACIÓN CON MPC

```
clear all
%% SIMULACIÓN CON MPC
max_u1=250; min_u1=50;
max_u2=6e4; min_u2=1e4;
max_y1=12; min_y1=0;
max_wQ=3.22e4; min_wQ=1e4;
Horizonte_horas=5;
Horizonte_en_pasos=Horizonte_horas*4;
refvar_2=10;
refvar_2=normalizar(refvar_2,max_y1,0,1);
Matrices_Jose_v3;
MPC_Jose;
%Cargar modelo y datos de caudal
load('modelo.mat');
load('wQ_fijo.mat');
n=modelo.n;n_y1=n(1);n_u1=n(2);n_u2=n(3);n_wQ=n(4);

%% Inicio simulacion
    Name=strcat('PRUEBA_MPC_5')
    u1=[];
    u1(1)=50;
    u1(2)=50;
```

```
    u2(1)=55338;
    [y1(1),y2(1),u1(1),u2(1),wQ(1)]=SimuladorWWTP(u1(1),u2(1),1,Name);
    j=2;

%% Avanzar simulación hasta tener datos
while length(y1)<max(n)+1
   [y1(j) , y2(j), u1(j), u2(j),wQ(j)]=SimuladorWWTP(u1(1),u2(1),0,Name);
    j=j+1;
end
    j=0;

%% BUCLE GLOBAL
for i=n_wQ:800
    for h=0:Horizonte_en_pasos-1
     precios_futuros(h+1)=calculaprecio(i+h,Horizonte_horas);
    end
    %% Inicio de vuelta
      xini=[normalizar(y1(end:-1:(end-n_y1+1)),max_y1,0,1)
      normalizar(u1(end-1:-1:(end-n_u1+1)),max_u1,min_u1,1)
      normalizar(u2(end-1:-1:(end-n_u2+1)),max_u2,min_u2,1)];
    uini=[normalizar(u1(end),max_u1,min_u1,1);
      normalizar(u2(end),max_u2,min_u2,1)];
    wQ_v=[normalizar(wQ(i:-1:(i-n_wQ+1)),max_wQ,min_wQ,1)
    normalizar(wQ(i:-1:(i-n_wQ+1)),max_wQ,min_wQ,1).^2
    normalizar(wQ(i:-1:(i-n_wQ+1)),max_wQ,min_wQ,1).^3];
[uk,diagnostics(i-n_wQ+1)] = controller{[xini'; refvar_2; precios_futuros';
wQ_v']};

%% Avanza un paso en el proceso de simulación del comportamiento de %la
depuradora usando el control predictivo
    if diagnostics(i-n_wQ+1)==0
        uk(1)=normalizar(uk(1),max_u1,min_u1,-1);
        uk(2)=normalizar(uk(2),max_u2,min_u2,-1);
    else
        uk(1)=u1(i-n_wQ+1-1);
        uk(2)=u2(i-n_wQ+1-1);
    end
[y1(i-n_wQ+1),y2(i-n_wQ+1),u1(i-n_wQ+1),u2(i-n_wQ+1),wQ(i-
 n_wQ+1)]=SimuladorWWTP(uk(1),uk(2),0,Name);

end
```

A5.13.- IDENTIFICACIÓN POR MÍNIMOS CUADRADOS Y OPTIMIZADOR

```
clear all
load('Resultado_rele_14das_escalones_todo2.mat')
wQ=normalizar0_1(wQ(1:end-1));
u1=normalizar0_1(Kla(2:end));
y1=normalizar0_1(S_nh(1:end-1));
u2=normalizar0_1(Q_out2(2:end));
y2=normalizar0_1(M_fangos(1:end-1));
t=t(1:end-1);

%% IDENT. MODELO
%delays
d_u1=1;
d_u2=0;
d_y1=0;
d_d=2;
d=[d_u1, d_u2,d_y1, d_d];
```

```
%dordenes
n_u1=5;
n_u2=3;
n_y1=5;
n_d=5;
n=[n_u1, n_u2,n_y1, n_d];
raiz=2;
y1=y1.^(1/raiz);
Y=y1(max(n)+1+max(d):end)';
N=length(y1);
X=[];
for i=max(n)+1+max(d):N
    X=[X;  1  y1(i-1-d_y1:-1:i-n_y1-d_y1)  u1(i-1-d_u1:-1:i-n_u1-d_u1)
u2(i-1-d_u2:-1:i-n_u2-d_u2)  wQ(i-1-d_d:-1:i-n_d-d_d)  wQ(i-1-d_d:-1:i-
n_d-d_d).^2 wQ(i-1-d_d:-1:i-n_d-d_d).^3];
end
theta=X\Y;
yh=y1;
for i=max(n)+1+max(d):N
    yh(i)=[1  yh(i-1-d_y1:-1:i-n_y1-d_y1)  u1(i-1-d_u1:-1:i-n_u1-d_u1)
u2(i-1-d_u2:-1:i-n_u2-d_u2)  wQ(i-1-d_d:-1:i-n_d-d_d)  wQ(i-1-d_d:-1:i-
n_d-d_d).^2 wQ(i-1-d_d:-1:i-n_d-d_d).^3]*theta;
end
%Exportación de modelo
modelo.theta=theta;
modelo.n=n;
save('modelo.mat','modelo')
figure, plot(y1.^raiz), hold on, plot(yh.^raiz,'r*')

[3*cov(y1.^raiz-yh.^raiz).^0.5 max(abs(y1.^raiz-yh.^raiz))]

%% VALIDACIÓN
load('Resultado_rele_14das_escalones_todo.mat')

wQ=normalizar0_1(wQ(1:end-1));
u1=normalizar0_1(Kla(2:end));
y1=normalizar0_1(S_nh(1:end-1));
u2=normalizar0_1(Q_out2(2:end));
y2=normalizar0_1(M_fangos(1:end-1));
t=t(1:end-1);
% figure, plot(t,y1,t,y2,t,wQ)

raiz=4;
y1=y1.^(1/raiz);
Y=y1(max(n)+1:end)';
N=length(y1);
X=[];
for i=max(n)+1:N
    X=[X; 1 y1(i-1:-1:i-n_y1) u1(i-1:-1:i-n_u1) u2(i-1:-1:i-n_u1) wQ(i-
1:-1:i-n_d) 0*wQ(i-1:-1:i-n_d).^2 wQ(i-1:-1:i-n_d).^3];
end
theta=X\Y;
yh=y1;
for i=max(n)+1:N
    yh(i)=[1  yh(i-1:-1:i-n_y1)  u1(i-1:-1:i-n_u1)  u2(i-1:-1:i-n_u1)
wQ(i-1:-1:i-n_d) 0*wQ(i-1:-1:i-n_d).^2 wQ(i-1:-1:i-n_d).^3]*theta;
end
figure, plot(y1.^raiz), hold on, plot(yh.^raiz)
```

```
[3*cov(y1.^raiz-yh.^raiz).^0.5 max(abs(y1.^raiz-yh.^raiz))]

%% con optimizador
load('Resultado_rele_14das_escalones_todo2.mat')

wQ=normalizar0_1(wQ(1:end-1));
u1=normalizar0_1(Kla(2:end));
y1=normalizar0_1(S_nh(1:end-1));
u2=normalizar0_1(Q_out2(2:end));
y2=normalizar0_1(M_fangos(1:end-1));
t=t(1:end-1);

%% IDENT. y1
raiz=2;
y1=y1.^(1/raiz);
Y=y1(max(n)+1+max(d):end)';
N=length(y1);
X=[];
for i=max(n)+1+max(d):N
    X=[X;  1  y1(i-1-d_y1:-1:i-n_y1-d_y1)  u1(i-1-d_u1:-1:i-n_u1-d_u1)
u2(i-1-d_u2:-1:i-n_u2-d_u2)  wQ(i-1-d_d:-1:i-n_d-d_d)  wQ(i-1-d_d:-1:i-
n_d-d_d).^2 wQ(i-1-d_d:-1:i-n_d-d_d).^3];
end
theta_var=sdpvar(size(X,2),1,'full');
t_var=sdpvar(1,1,'full');
restricciones=[-t_var<=(Y-X*theta_var)<=t_var,t_var>=0,...
-0.95<=theta_var(2:n_y1+1)<=0.95];
Objetivo=t_var+0.3*norm(theta_var);

opciones = sdpsettings('solver','sedumi');
solucion = optimize(restricciones,Objetivo,opciones);
theta=value(theta_var);
value(t_var)
yh=y1;
for i=max(n)+1+max(d):N
    yh(i)=[1  yh(i-1-d_y1:-1:i-n_y1-d_y1)  u1(i-1-d_u1:-1:i-n_u1-d_u1)
u2(i-1-d_u2:-1:i-n_u2-d_u2)  wQ(i-1-d_d:-1:i-n_d-d_d)  wQ(i-1-d_d:-1:i-
n_d-d_d).^2 wQ(i-1-d_d:-1:i-n_d-d_d).^3]*theta;
end

figure, plot(y1.^raiz), hold on, plot(yh.^raiz)
[3*cov(y1.^raiz-yh.^raiz).^0.5 max(abs(y1.^raiz-yh.^raiz))]
```

A5.14.- FUNCIÓN DE RELÉ "ALEATORIO"

```
function [u_new act]=ReleRand(ref1,y,u_max,u_min,act,u)

    if (ref1+0.5)<y
        if act
            u_new=u;
        else
            u_new=u_max*(rand*0.15+0.9);
            act=1;
        end
    elseif (ref1-0.5)>y
        if act
            u_new=u_min*(rand*0.15+0.9);
            act=0;
        else
            u_new=u;
        end
```

```
    else
        u_new=u;
        act=act;
    end
```

A5.15.- FUNCIÓN DE CÁLCULO DE PRECIO ELECTRICIDAD

```
function precio=calculaprecio(num_pasos,incremento)
tiempodia= floor((num_pasos*incremento-floor(num_pasos*incremento))*24);
switch tiempodia
 case 0
 precio=0.087564;
 case 1
 precio=0.087564;
 case 2
 precio=0.087564;
 case 3
 precio=0.087564;
 case 4
 precio=0.087564;
 case 5
  precio=0.087564;
 case 6
 precio=0.087564;
 case 7
 precio=0.087564;
 case 8
 precio=0.130477;
 case 9
 precio=0.130477;
 case 10
 precio=0.146984;
 case 11
 precio=0.146984;
 case 12
 precio=0.146984;
 case 13
 precio=0.130477;
 case 14
 precio=0.130477;
 case 15
 precio=0.130477;
 case 16
 precio=0.130477;
 case 17
 precio=0.130477;
 case 18
 precio=0.146984;
 case 19
 precio=0.146984;
 case 20
 precio=0.146984;
 case 21
 precio=0.130477;
 case 22
 precio=0.130477;
 case 23
 precio=0.130477;
end
```

A5.16.- ALGORITMO DE CONTROL PREDICTIVO

```
nu=2; % Número de u a calcular (número de entradas)
u =
sdpvar(repmat(nu,1,Horizonte_en_pasos),repmat(1,1,Horizonte_en_pasos))
;
refvar=sdpvar(1,1);
tam_modelo=size(M);
tam_Q=size(Q);
objective=0;
constraints=[];
A=M;
x0 = sdpvar(tam_modelo(2),1);
wQ_var=sdpvar(tam_Q(2),1);

precio_sdpvar=sdpvar(Horizonte_en_pasos,1);
x=x0;
for k = 1:Horizonte_en_pasos
   x = A*x + B(:,1:2)*u{k}+Q*wQ_var+C;
   objective =objective+precio_sdpvar(k)*u{k}(1);

     if k==1
         constraints = [constraints, -500<=(u{k}(1)-u0(1))<=500, -
0.5<=(u{k}(2)-u0(2))<=0.5];

     else
         constraints = [constraints, -500<=(u{k}(1)-u{k-1}(1))<=500, -
0.5<=(u{k}(2)-u{k-1}(2))<=0.5];

     end
  constraints = [constraints,
0<=((u{k}(1)))<=1,0<=((u{k}(2)))<=1,x(1)<=(refvar)^0.5];
  if k>1
      constraints = [constraints, -0.1<=(u{k}-u{k-1})<=0.1];
  end
  ops = sdpsettings('verbose',0,'solver','SEDUMI');
%MOSEK va, SEDUMI va pero lento, CUTSDP va mediolento, fmincon
lentísimo

  controller = optimizer(constraints, objective, ops, [x0; refvar;
precio_sdpvar; wQ_var],u{1});

en
```

III. PLIEGO DE CONDICIONES

ÍNDICE DE CAPÍTULO

ÍNDICES DE TABLAS Y FIGURAS

III.1.- INTRODUCCIÓN

Dados el alcance y la naturaleza del proyecto, así como las características técnicas del mismo, el Pliego de Condiciones se reduce a los siguientes apartados.

III.2.- REQUISITOS

Para poder replicar experimentos o validar los cálculos recogidos en este proyecto. Es necesario tener en cuenta las siguientes premisas:

1. Los experimentos realizados y por tanto los resultados obtenidos se basan en el texto y datos recogidos en el BSM1.
2. Para alcanzar los resultados exactos, se deben cumplir los criterios recogidos en los apartados de este Pliego de Condiciones.
3. El correcto funcionamiento de las aplicaciones viene condicionado por el Material Utilizado y por lo tanto es necesario cumplir con las especificaciones técnicas que se exigen.
4. El código utilizado (recogido en el **ANEXO 5: CÓDIGO UTILIZADO**) debe ser respetado y utilizado según se indica en los comentarios del mismo.
5. Para que las aplicaciones distribuibles funcionen según lo previsto, es indispensable la lectura de los Manuales recogidos en este Pliego.

III.3.- MATERIALES UTILIZADOS

III.3.1.- HARDWARE

Para que el proyecto se pueda llevar a cabo el hardware ha de cumplir con los requisitos de instalación de los softwares de cálculo y modelado utilizados.

Requisitos para MATLAB 2019a:

SISTEMA OPERATIVO	PROCESADOR	DISCO
Windows 10 Windows 7	**Mínimo:** - Cualquier Intel o AMD de x86-64 **Recomendado:** - X86-64 con 4 núcleos y AVX	**Mínimo:** - 2.9GB de HDD **Recomendado:** - SSD

RAM	GRÁFICOS
Mínimo: - 4 GB **Recomendado:** - 8 GB	No hay especificaciones requeridas.

Tabla I.- Requisitos de hardware

Requisitos para la conexión con Tornado:

Puesto que la licencia de **WEST** con la que se cuenta es para la versión x32. Se necesita que la versión de **Matlab** que ejecuta el simulador también sea de 32.

III.3.2.- SOFTWARE

En la realización del proyecto se han utilizado dos versiones de **Matlab**, por un lado, la versión R2013b para la construcción del simulador y del algoritmo de control, y por otro, la versión R2019a para el diseño y empaquetamiento de la aplicación. En cuanto a WEST, se ha trabajado con la versión 2016 del paquete **MIKE by DHI.**

Esto quiere decir que, si se utilizan versiones diferentes de los softwares previamente nombrados, no se podría asegurar el correcto funcionamiento de los algoritmos y programas desarrollados en este proyecto.

III.4.- CRITERIOS DE TOMA DE DATOS Y VALIDACIÓN

III.4.1.- UNIDADES DEL PROYECTO

Uno de los puntos que puede generar más problemas y posibles puntos de fallos son los errores en las unidades de las variables. Por lo tanto, en este apartado, se recogen las unidades en función de la naturaleza de la variable para asegurar que no se producen mal entendidos o confusiones.

TIPO DE VARIABLE	UNIDADES
Volumen	Metros cúbicos - m^3
Tiempo	Días – d
Caudal	Metros cúbicos por día - m^3/día - m^3/d
Concentración	Gramos por metro cúbico – g/ m^3
Especiales	- K_La [1/día] - Precio electricidad [€/kWh o €/MWh] - Precipitaciones [mm de lluvia]

Tabla II.- Unidades en función del tipo de variable

III.4.2.- DATOS DEL SIMULADOR

Para asegurar que el simulador funciona correctamente, es indispensable verificar los siguientes puntos:

- El **formato de los datos de entrada** es el adecuado en cuanto al tipo de fichero (.txt, .xml, .csv …) y a la estructura interna del mismo (nº de columnas, orden de columnas, nº de datos…).
- Las **unidades de los datos de entrada.** Se debe verificar que las variables de entrada están en las unidades correctas
- Los **permisos de escritura y lectura**. Tanto las carpetas desde las que se leen los datos de entrada como el destino de los resultados deben ser accesibles para el simulador.
- El **tratamiento de las salidas**. Los ficheros de salida cuentan con un formato y estructura establecidos y para interpretarlos se debe conocer y respetar.

III.4.3.- DATOS DE LA EDAR REAL

Aunque no se haya materializado la implantación de la aplicación en una estación real, es importante gestionar la recogida de datos. Siendo indispensable la correcta instalación y calibración de los sensores.

La lista de los principales sensores a instalar para poder asegurar el correcto funcionamiento del algoritmo de control son los siguientes:

- Caudalímetro a la **entrada de la planta**.
- Sonda de concentración del **Amonio a la salida**.
- Caudalímetro de **fangos a la salida**.

También sería recomendable, contar con medidores de potencia para realizar una correcta monitorización y cálculo del consumo de cada equipo.

III.5.- MANTENIMIENTO Y MODIFICACIONES DE LA APLICACIÓN

Al distribuirse la aplicación final al usuario, es indispensable establecer el protocolo de actuación frente a los procesos de mantenimiento y modificaciones de la aplicación. Las condiciones se recogen a continuación:

- El mantenimiento de los equipos corre a cargo del cliente, siendo su deber mantenerlos en correcto estado. Por lo tanto, si la aplicación o el control fallan por motivos relacionados con el mantenimiento de los equipos no son responsabilidad de los desarrolladores.
- Los fallos derivados del uso de la aplicación solamente serán considerados como fallos de programación si se han respetados todos los puntos recogidos en el pliego de condiciones.
- La aplicación únicamente puede ser modificada por los desarrolladores autorizados y deberán ser notificados y documentados correctamente.
- En caso de realizarse mejoras o aparecer actualizaciones de la propia aplicación se asegura que el cliente será notificado y se le hará llegar la nueva versión.
- Si existen problemas de incompatibilidad con futuras actualizaciones de los sistemas operativos o aplicaciones, se estudiará la posibilidad de adaptar la aplicación a las nuevas condiciones y si no fuera posible será responsabilidad del cliente no actualizar los equipos.
- Las modificaciones o variaciones realizadas sobre la aplicación fuera del acuerdo establecido serán motivo de pérdida de derecho al uso de la misma.
- La distribución ilegal de la aplicación será perseguida y castigada con las herramientas legales pertinentes.

III.6.- MANUAL DE INSTALACIÓN DE APLICACIÓN DE CONTROL

Esta guía de instalación proporciona instrucciones sobre como instalar la aplicación **JCR_EDAR_Solutions**, además de contar con una pequeña guía sobre la misma.

PASO 1: Adquirir vía Internet o mediante USB o CD la aplicación.

PASO 2: Descomprimir el fichero que contiene el ejecutable "JCR_EDAR _Solution.exe"

PASO 3: Ejecutar el archivo en modo administrador.

PASO 4: Seguir los pasos marcados en el instalador. Pulsar siguiente ("Next").

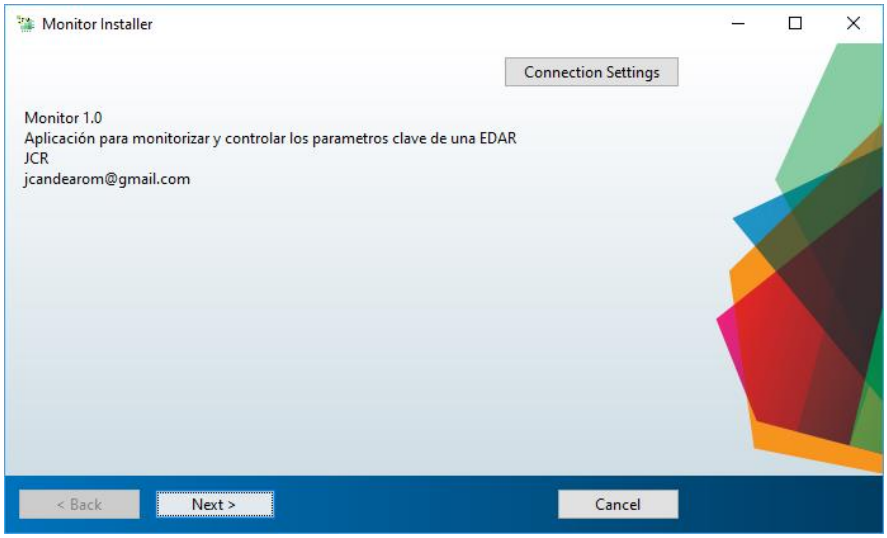


Figura I.-Recorte PASO 1

PASO 5: Esperar a que se complete la preparación de ficheros.

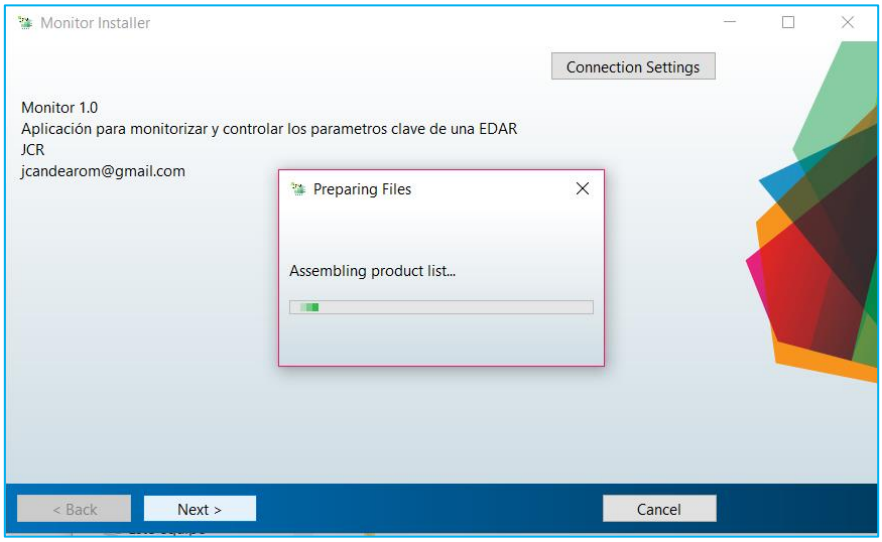


Figura II.- Recorte PASO 2

- Seleccionar la carpeta en la que se va a instalar y elegir sí se quiere añadir un acceso directo al escritorio.

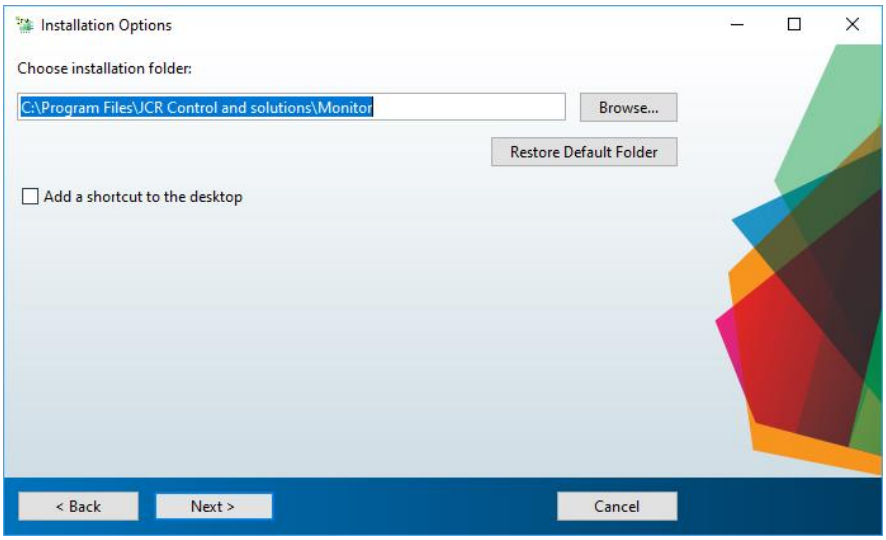


Figura III.- Recorte Selección Carpeta

- Instalar "MATLAB Runtime"

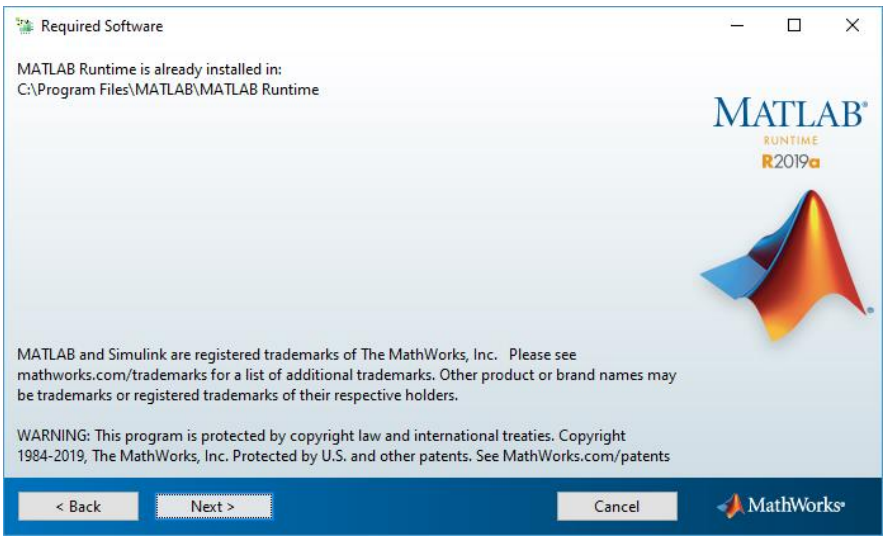


Figura IV.- Recorte Instalación Runtime

- Comprobar verificación y pulsar ***INSTALL***

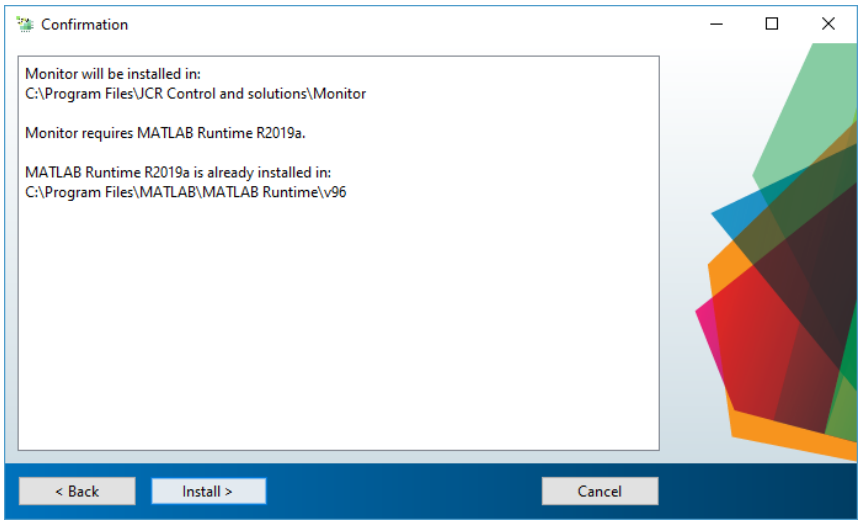


Figura V.- Recorte Install

- Esperar a que se complete la instalación.

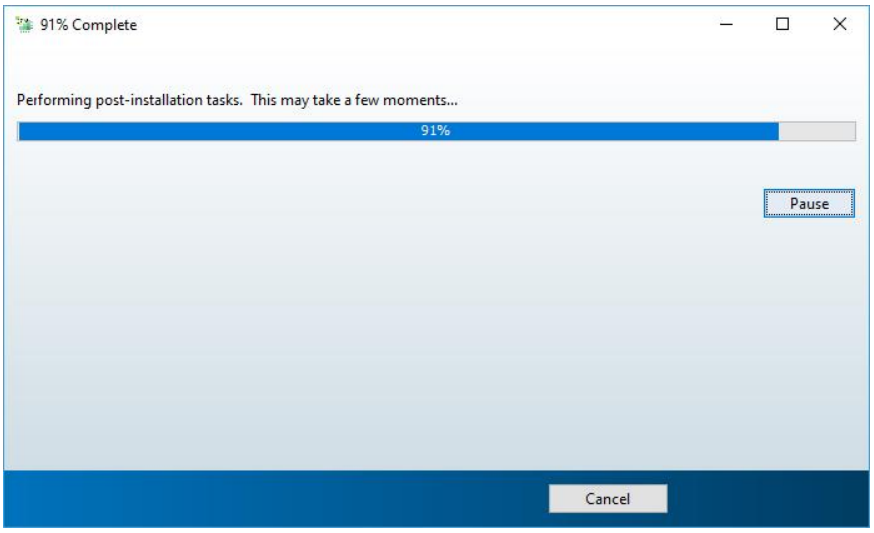


Figura VI.- Progreso de la instalación

- Pulsar ***FINISH*** para salir del proceso de instalación.

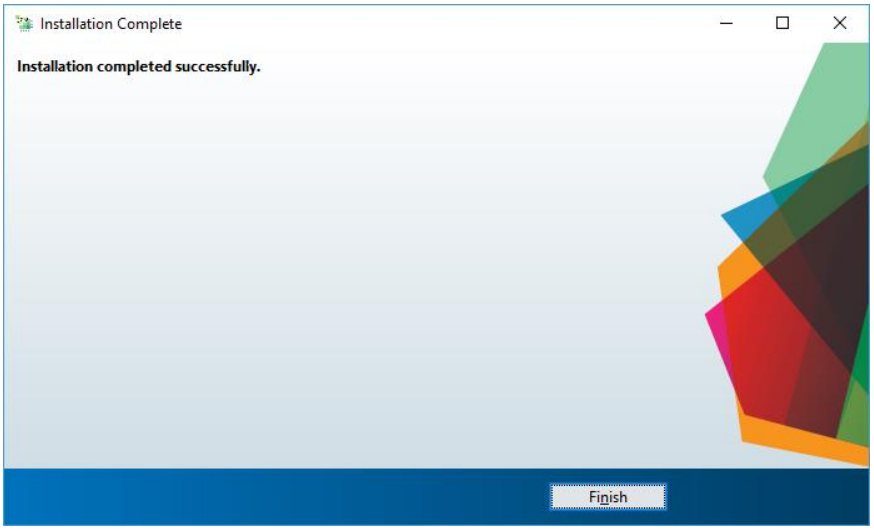


Figura VII.- Pantalla de Finish

PASO 6: Buscar el acceso directo y hacer doble clic, entonces se desplegará la aplicación como cualquier otra propia del ordenador.

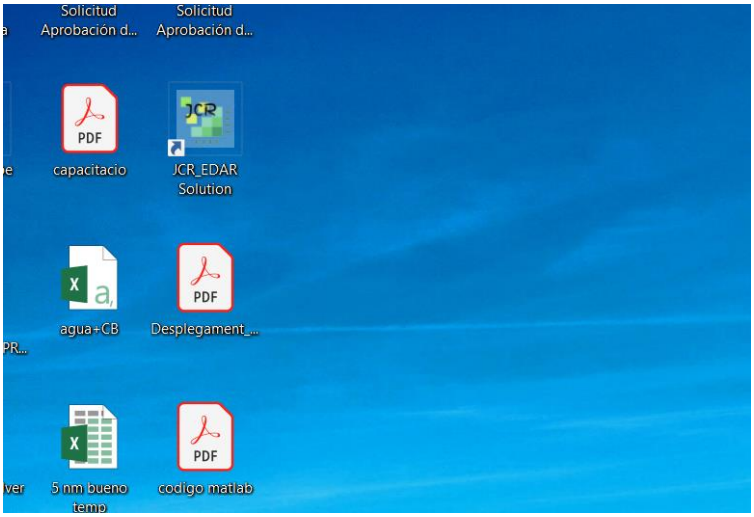


Figura VIII.- Recorte de acceso directo

PASO 7: Una vez cargada la aplicación es totalmente funcional.

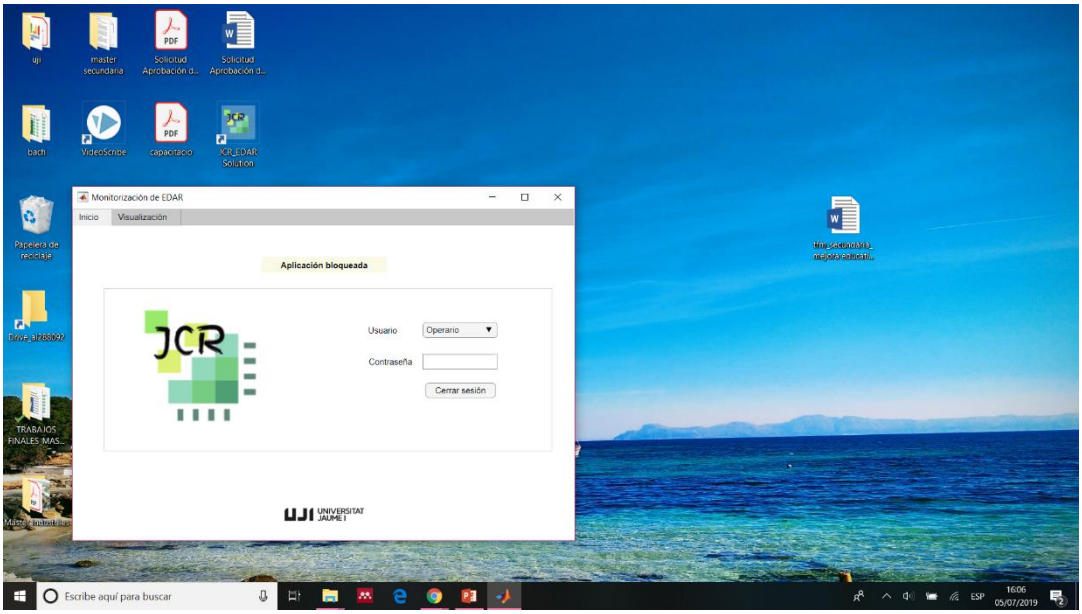


Figura IX.- Recorte despliegue de aplicación

III.7.- MANUAL DE APLICACIÓN DE IDENTIFICACIÓN

En este manual se explica el proceso a seguir para identificar el sistema y exportar el modelo resultante.

PASO 1: Instalación de la aplicación siguiendo los pasos del MANUAL DE INSTALACIÓN

PASO 2: Ejecutar la aplicación, a continuación, se abre la siguiente ventana.

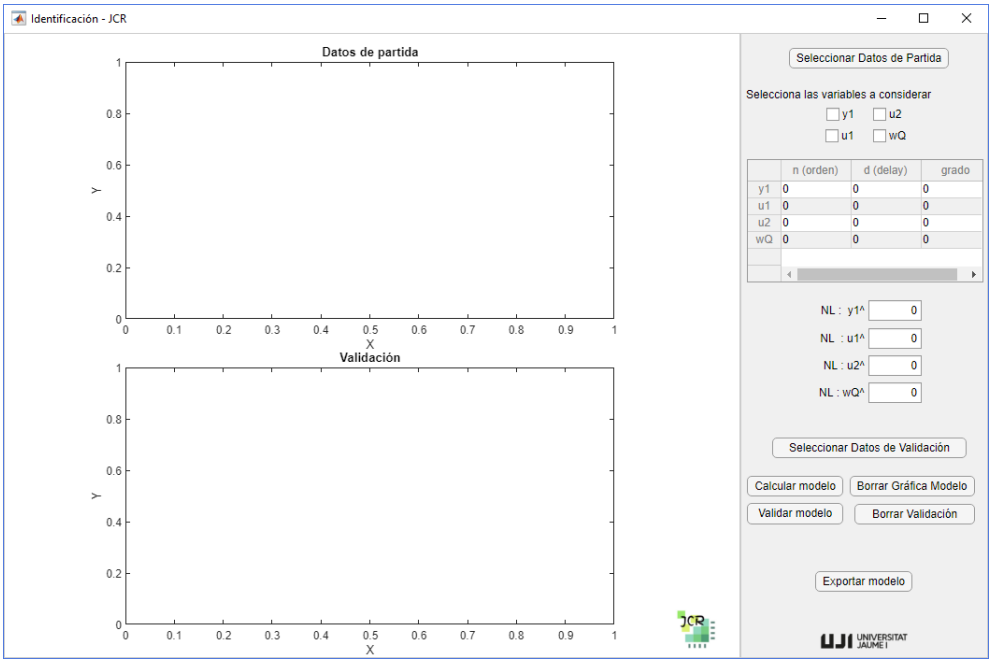


Figura X.- Vista general de la aplicación de IDENTIFICACIÓN

PASO 3: Se seleccionan los Datos de Partida, pulsando el botón “Seleccionar Datos de Partida”. Al pulsar el botón, se abre el siguiente cuadro de dialogo que sirve para seleccionar el fichero con los datos de la excitación.

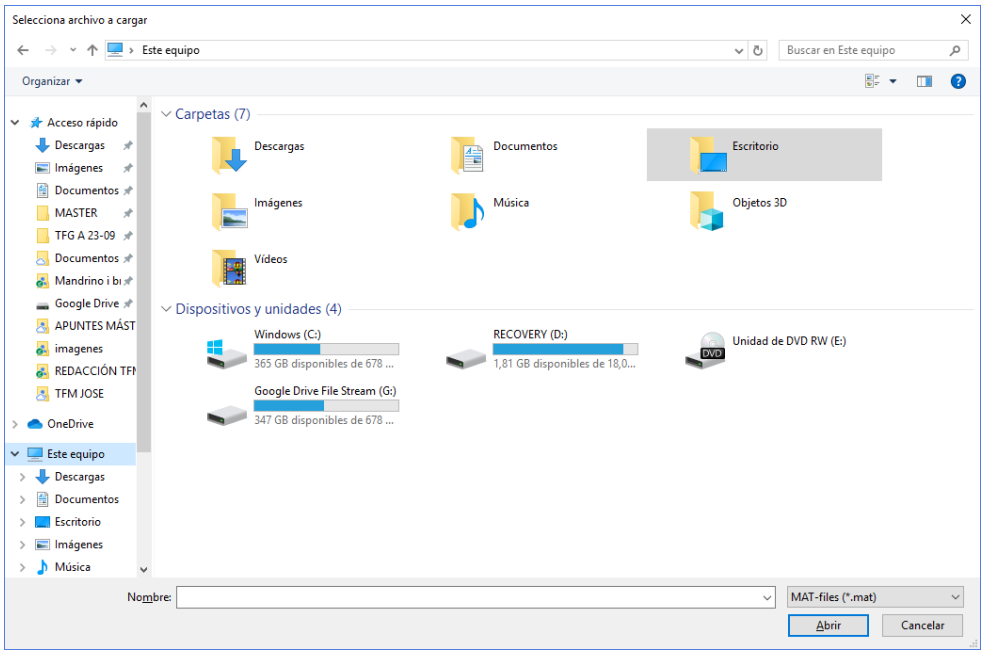


Figura XI.- Ventana de selección de los datos

PASO 3: Se seleccionan las variables que se quieren tener en cuenta en la construcción del modelo. Para hacer esto, en la aplicación aparecen una serie de *Check-boxes* que se activan al pulsar.

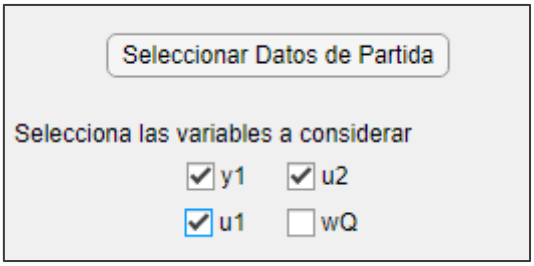
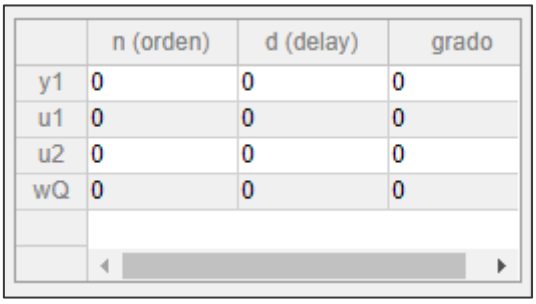


Figura XII.- Botones y check boxes

PASO 4: En la tabla que aparece justo debajo de los *checks* se introducen los datos de orden, *delay* y grado en el modelo.



	n (orden)	d (delay)	grado
y1	0	0	0
u1	0	0	0
u2	0	0	0
wQ	0	0	0

Figura XIII.- Selección ordenes, delays y grado

PASO 5: Justo debajo de esta tabla, aparecen unos cuadros editables que permiten establecer las no linealidades a la entrada y salida del modelo lineal.

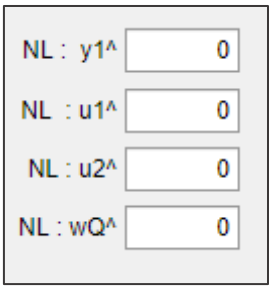


Figura XIV.- Introducción de NL

PASO 6: En los botones que aparecen a continuación, se pueden realizar diferentes acciones.

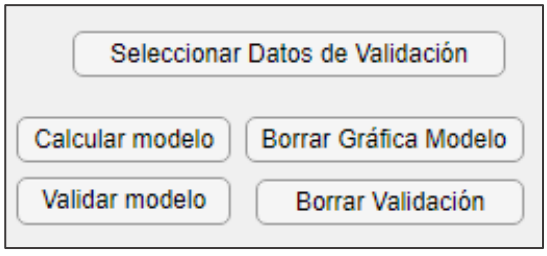


Figura XV.- Botones de acción

Calcular modelo – Calcula los coeficientes del modelo y representan los resultados.
Borrar Gráfica Modelo – Vacía los ejes de la parte superior (Datos de Partida).
Seleccionar Datos de Validación – Abre una ventana de dialogo que permite seleccionar los Datos con los que validar el modelo.
Validar modelo – Carga el modelo y lo representa para los datos de Validación.
Borrar Validación – Vacía los ejes de la parte inferior (Validación).

PASO 7: Se modifican y se repite el proceso hasta que se consideré que el modelo es correcto.

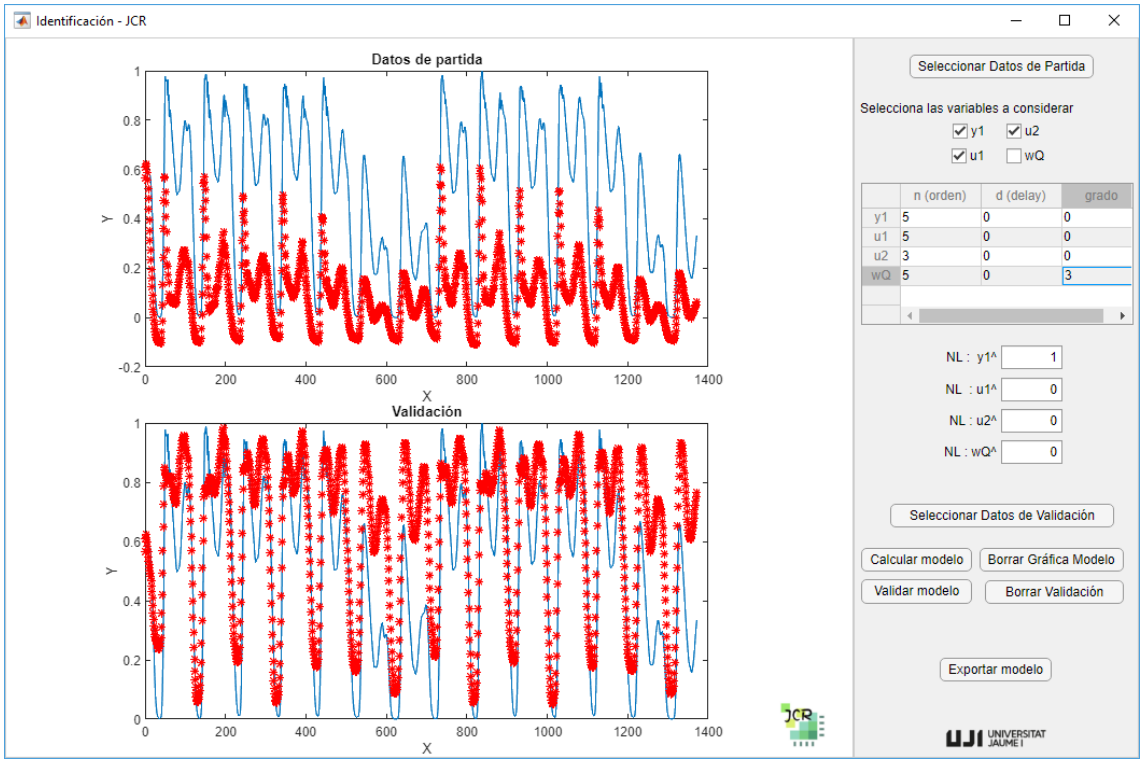


Figura XVI.- Imagen general de modelo y validación

PASO 8: Al considerar válidos los resultados representados. Se pulsa el botón de Exportar modelo y se crea una variable con el nombre de "modelo.mat" con los datos más importantes de este.

Figura XVII.- Botón de exportación

IV. PRESUPUESTO

ÍNDICE DE CAPÍTULO

IV.1.- INTRODUCCIÓN

En este documento se recopilan los datos detallados de todos los gastos y costes relacionados con la gestión, el inicio y el desarrollo de este proyecto. Para mayor claridad el capítulo se divide en varias partes en las que se realiza un desglose exhaustivo de los precios unitarios de cada uno de los elementos.

Por último, en el apartado final se calcula el presupuesto completo contabilizando tanto los costes como sus impuestos y la parte proporcional en concepto de **Beneficio Industrial** y **Gastos Generales.**

Los datos recogidos en esta parte del proyecto se consideran a todos hechos como correctos y se utilizan en la parte de viabilidad económica para calcular los índices que se consideren oportunos.

IV.2.- PRESUPUESTO MATERIAL Y EQUIPOS

DESCRIPCIÓN DEL PRESUPUESTO
En este presupuesto se presentan los costes derivados de la ejecución del proyecto.

PRESUPUESTO DE MATERIAL Y EQUIPOS

UNIDAD	CONCEPTO	CANTIDAD	PRECIO UNIDAD (€)	PRECIO(€)
ud	*Licencia completa de MATLAB 2019a*	*1*	*600,00*	*600,00*
meses	*Licencia pack URBAN UTILITIES (WEST)*	*6*	*750,00*	*4500,00*
ud	*Ordenador de sobremesa*	*1*	*950,00*	*950,00*

TOTAL (€)
6050,00

Figura. VI.- Presupuesto de material y equipos

IV.3.- PRESUPUESTO RECURSOS HUMANOS

DESCRIPCIÓN DEL PRESUPUESTO
En este presupuesto se presentan los costes derivados de los recursos humanos que intervienen del proyecto.

PRESUPUESTO DE RECURSOS HUMANOS

UNIDAD	CONCEPTO	CANTIDAD	PRECIO UNIDAD (€)	PRECIO(€)
h	*Desarrollo del proyecto (ingeniería)*	*350*	*30,00*	*10500,00*

TOTAL (€)
10500,00

Figura. VII.- Presupuesto recursos humanos

IV.4.- PRESUPUESTO FINAL

PRESUPUESTO DE MATERIAL Y EQUIPOS	***6050,00 €***
PRESUPUESTO DE RECURSOS HUMANOS	***10500,00 €***
PRESUPUESTO DESARROLLO DEL PROYECTO	***16550,00 €***
13% GASTOS GENERALES	*2151,50 €*
6% BENEFICIO INDUSTRIAL	*933,00 €*
PRESUPUESTO DE EJECUCIÓN	***19695,50 €***
21% IVA.	*4135,80 €*

TOTAL (€)
23830,30

Figura. VIII.- Presupuesto final

El presupuesto final es de **VEINTITRÉS MIL OCHOCIENTOS TREINTA EUROS CON TREINTA CÉNTIMOS.**